HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

INSTITUTE FOR INTERNET
TECHNOLOGIES AND APPLICATIONS

Bachelorarbeit, Abteilung Informatik

Kinect im Gesundheitswesen

Hochschule für Technik Rapperswil

Frühlingssemester 2013

Autoren:	Pascal Roman Artho & Rebekka Zahler
Betreuer:	Prof. Dr. Josef M. Joller
Experte:	Matthias Lips
Projektpartner:	ITA: Institut für Internet-Technologien und -Anwendungen
Arbeitsperiode:	18. Februar 2013 bis 14. Juni 2013
Arbeitsumfang:	360 Stunden, 12 ECTS pro Student

Danksagung

Prof. Dr. Josef M. Joller für die Ideengebung und seine Unterstützung während der Arbeit.

Mirco Widmer, Cyrill Lam und Paul Fäh für das Testen der Applikation.

Anna Artho und Michael Schnyder für das Korrekturlesen der Bachelorarbeit.

Freunden und Familienangehörigen für ihre Unterstützung und Geduld während dieser wichtigen Zeit.

Abstract

Abteilung	Informatik
Name[n] der Studierenden	Pascal Roman Artho Rebekka Zahler
Studienjahr	Frühlingssemester 2013
Titel der Studienarbeit	Kinect im Gesundheitswesen
Examinatorin / Examinator	Prof. Dr. Josef M. Joller
Experte	Matthias Lips
Themengebiet	Internet-Technologien und -Anwendungen
Projektpartner	ITA: Institut für Internet-Technologien und -Anwendungen
Institut	ITA: Institut für Internet-Technologien und -Anwendungen

Ausgangslage

Für viele Menschen ist es schwierig, eine Therapieübung vor einem Therapeuten oder Arzt durchzuführen. Sie stehen unter ständiger Beobachtung. Oft ist die Hemmung und die Angst zu scheitern grösser, als die Hoffnung auf Erfolg. Hinzu kommt, dass insbesondere bei langwierigen Therapien die Kosten für die Anwesenheit eines Therapeuten in die Höhe schnellen.

Vorgehen / Technologien

Die in der Ausgangslage erwähnten Überlegungen waren die Grundlage für diese Bachelorarbeit. Durch verschiedene Recherchen wurde versucht herauszufinden, welche Lösungen in diesem Bereich bereits bestehen und welche Art von Applikationen zum Erfolg führen. In erster Linie wurden Lösungen mit der Kinect, eine von Microsoft entwickelte Kamera, gesucht. Es handelt sich um ein kostengünstiges und einfach handhabbares Produkt.

Schnell zeigte sich, dass im Internet nur begrenzt Lösungen mit einer Kinect verfügbar sind. Der Bewegungsvorgang ist dabei in den meisten Fällen fest vorgegeben, sodass es sinnvoll ist, eine eigene Applikation zu entwickeln.

Ergebnis

Im Rahmen dieser Bachelorarbeit wurde eine Applikation namens «Medical Kinect» entwickelt. Sie ermöglicht es dem Benutzer, mit der Kinect Übungen aufzuzeichnen und sie durchzuführen. Dabei wird anhand von den Positionen der erkannten Gelenke überprüft, ob die Übung korrekt durchgeführt wird.

Da die Positionen der Gelenke bei jedem Mensch verschieden sind, muss der Körper zuerst kalibriert und damit ausgemessen werden. Danach können die einzelnen Schritte der Übung aufgezeichnet werden. Zum Schluss wird ausgewählt, auf welche Körperteile und Achsen-Richtungen geachtet werden soll. Aus all diesen Informationen wird eine XML-Datei generiert. Bevor eine Übung durchgeführt werden kann, muss wiederum der Körper kalibriert werden. So können die Positionen der Gelenke des aktuellen Benutzers mit den gespeicherten Werten verglichen werden.

Aufgrund des günstigen Preises einer Kinect, kann das entwickelte Programm auch im Alltag sehr gut eingesetzt werden. Menschen, welche viel Sitzen und sich selten bewegen, können mit der Kinect und der entwickelten Applikation gezielt Übungen durchführen. Dadurch, dass die Übungen in einer XML-Datei gespeichert sind, können sie beliebig mit anderen Benutzer der Applikation, beispielsweise Patienten, ausgetauscht werden.

Aufgabenstellung

Studiengang:	Informatik (I)
Semester:	Frühlingssemester 2013 (18. Februar 2013 bis 14. Juni 2013)
Verantwortlicher:	Prof. Dr. Josef M. Joller
Gegenleser:	Prof. Dr. Olaf Zimmermann
Experte:	Matthias Lips
Projektpartner:	ITA: Institut für Internet-Technologien und -Anwendungen

Ausgangslage

Behinderte Menschen und Patienten, welche beispielsweise einen Hirnschlag erlitten haben, lernen in Therapien einfache Bewegungen durchzuführen. Sie sind dabei unter ständiger Beobachtung eines Therapeuten, welcher die Übung vorzeigt und überprüft, ob diese sauber durchgeführt wird. Viele haben Hemmungen davor, Übungen Drittpersonen vorzuzeigen, da ihr Bewegungsvorhaben scheitern könnte. Die Motivation die Übungen durchzuführen kann daher sehr schnell sinken, wodurch eine schnelle Rehabilitation in weite Ferne rücken kann.

Ziel

Im Rahmen der Bachelorarbeit wird untersucht, inwiefern Therapien für behinderte Menschen und Patienten durch den Einsatz von Kinect unterstützt werden können. Mit diesen Erkenntnissen soll eine Applikation entwickelt werden, welche genau auf die Bedürfnisse der Patienten abgestimmt ist. Die Anwendung soll Bewegungen aufnehmen können und in passender Form mit den Bewegungen des Patienten verglichen und ausgewertet werden. Die Kinect prüft, ob der Patient die Bewegung richtig ausführt. Zusätzlich soll die Bedienung möglichst benutzerfreundlich sein, damit der Patient diese trotz Bewegungseinschränkung bedienen kann.

Rapperswil, den 10. Juni 2013 .....

Betreuer: Prof. Dr. Josef M. Joller

Erklärung zur Urheberschaft

Die vorliegende Arbeit basiert auf Ideen, Arbeitsleistungen, Hilfestellungen und Beiträgen gemäss folgender Aufstellung:

Gegenstand, Leistung	**Person**	**Funktion**
Kapitel 1, 4, 6, Schlussfolgerung, Anhänge	Pascal Roman Artho	Autor der Arbeit
Kapitel 2, 3, 5, Abstract, Management Summary, Anhänge	Rebekka Zahler	Autorin der Arbeit
Ideengeber, Betreuung	Prof. Dr. Josef M. Joller	Betreuer
Latex Vorlage	Florian Bentele	Student
Hilfestellung	Prof. Dr. Hansjörg Huser	
Usability Test	Mirco Widmer	
Usability Test	Cyrill Lam	
Usability Test	Paul Fäh	
Korrektur	Anna Artho	Lektorat
Korrektur	Michael Schnyder	Lektorat

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit gemäss obiger Zusammenstellung selber und ohne weitere fremde Hilfe durchgeführt habe,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Rapperswil, den 12.06.13 ..... P. Artho .....
Student: Pascal Roman Artho

Rapperswil, den 12.06.13 ..... R. Zahler .....
Studentin: Rebekka Zahler

Vereinbarung zur Verwendung und Weiterentwicklung der Arbeit

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Bachelorarbeit «Kinect im Gesundheitswesen» von Pascal Roman Artho und Rebekka Zahler unter der Betreuung von Prof. Dr. Josef M. Joller (für die Arbeit verantwortlicher Professor) geregelt.

2. Urheberrecht

Die Urheberrechte stehen der Studentin / dem Student zu.

3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von allen an der Arbeit beteiligten Parteien, daher von den Studenten, welche die Arbeit verfasst haben, vom verantwortlichen Professor sowie vom Industriepartner verwendet und weiterentwickelt werden. Die Namensnennung der beteiligten Parteien ist bei der Weiterverwendung erwünscht.

Rapperswil, den 10. Juni 2013 ..... P. Artho

Student: Pascal Roman Artho

Rapperswil, den 10. Juni 2013 ..... R. Z.

Studentin: Rebekka Zahler

Rapperswil, den 20. Juni 2013 ..... Joller

Betreuer: Prof. Dr. Josef M. Joller

Management Summary

Ausgangslage

Für viele Menschen mit Bewegungseinschränkung ist es schwierig, Therapieübungen Drittpersonen vorzuzeigen. Dafür ist oftmals die Hemmschwelle zu gross. Auf dem bisherigen Stand der Technik gibt es auch kaum Möglichkeiten, dass die Patienten mit Hilfe von technischen Geräten, welche die Ausführung überwachen kann, Übungen ausführen können.

Diese Lücke in der Technik soll ausgenutzt werden, um eine Applikation zu entwickeln, welche genau diese Zielpersonen erreicht. Hierfür wird die von Microsoft entwickelte Kamera, Kinect für Xbox 360, verwendet (siehe Abbildung 1). Mit dieser wird versucht, eine Lösung zu finden, mit welcher Personen einfache Übungen selber erstellen und bereits erstellte Aufgaben ausführen können.

Abbildung 1: Kinect für Xbox 360 [20]

Vorgehen

In einem ersten Schritt wurde versucht durch Recherchen herauszufinden, welche Applikationen für den Einsatz in der Medizin bereits bestehen und in welchen Themenbereichen. Für dieses Thema gibt es bisher kaum Literatur, deshalb wurde für die Recherche ausschliesslich das Internet benutzt. Durch die dadurch erworbenen Kenntnisse wurde in einem weiteren Schritt die Kinect kennengelernt und Versuche durchgeführt. Dies beschränkte sich vor allem darauf, ein «Skelett» anzeigen zu können und mit diesem zu arbeiten. Diese Versuche sind die Grundsteine dieser Bachelorarbeit.

Die erste Applikation die daraus entstand, konnte die grundlegenden Funktionen, wie das Auswählen und Durchführen von Übungen sowie Konfigurationen vornehmen. Bei

der Durchführung wurde anhand von Winkelberechnungen des Skeletts berechnet, ob die Übung richtig durchgeführt wurde. Allerdings wurde schnell erkannt, dass diese vorhandenen Funktionen nicht ausreichend sind. Wenn eine neue Übung erstellt werden soll, sind zwingend Programmierkenntnisse erforderlich, was den Einsatz der Applikation einschränkt. Deshalb wurde versucht eine weitere Funktion zu implementieren, mit welcher eine Person selber eine Übung erstellen kann.

Aufgrund dieser erweiterten Funktionalität müssen die Übungen generisch erstellt werden können. Die Übungen sollen nicht mehr fest programmiert werden. Gleichzeitig bedeutet das, dass nicht mehr die Winkel von einzelnen Gelenkpositionen für die Überprüfung ausschlaggebend sind, sondern die relative Position der jeweiligen Gelenke.

Ergebnisse

Im Verlauf der Bachelorarbeit wurde eine Vollbild-Applikation namens «Medical Kinect» entwickelt (siehe Abbildung 2). Dank der Verwendung der Kinect kann die Bedienung vollumfänglich durch Handzeichen übernommen werden. Die Applikation bietet dem Anwender die Möglichkeit, Therapieübungen auszuführen. Durch die Kinect wird überprüft, ob die Ausführung richtig durchgeführt wurde. Des Weiteren besteht die Möglichkeit selber Übungen zu erstellen und diese auszuführen oder auch mit anderen Anwendern auszutauschen.

Abbildung 2: Screenshot: Übungsdurchführung mit der «Medical Kinect» Applikation

Durch Usability Tests mit möglichen Anwendern zu verschiedenen Zeitpunkten konnte die Applikation immer weiter angepasst und den Wünschen entsprechend weiterentwickelt werden. Es ist ein benutzerfreundliches Programm entstanden, welches einfach zu bedienen ist.

Ausblick

Durch den tiefen Kostenpunkt einer Kinect, besteht durchaus die Möglichkeit, dass diese Applikation und vor allem auch die Idee, Zukunft haben kann. Mit der aktuellen Version könnte beispielsweise ein Therapiezentrum mit geringem Aufwand, allgemeine Übungsaufgaben aufnehmen. Diese könnten den Patienten direkt oder zentralisiert über einen Webserver übermittelt werden.

Zum jetzigen Zeitpunkt ist jedoch nicht klar, ob und zu welchem Anteil Versicherungsgesellschaften wie Krankenkassen oder Unfallversicherungen für die neuen Therapiekosten aufkommen. Das birgt die Gefahr, dass Therapeuten weiterhin und vollumfänglich auf die herkömmlichen Therapien setzen.

Generell ist anzunehmen, dass Microsoft an der Kinect weiterentwickelt, um noch präzisere Werte zu liefern. Diese Annahme wird durch die Medienmitteilung vom Mai 2013 von Microsoft bekräftigt. Sie hat die neue «Xbox One» mit der dazugehörenden Kinect vorgestellt. Die Kinect der zweiten Generation soll bis zu sechs Personen gleichzeitig erfassen und noch mehr Punkte am menschlichen Körper wahrnehmen. Wie das Internetportal «www.golem.de» in einem Artikel berichtet, wird der Sensor für den PC voraussichtlich nächstes Jahr auf den Markt kommen. [7]

Inhaltsverzeichnis

Kapitel 1

Einleitung

Für viele Patienten ist es unangenehm eine Bewegungstherapieübung vor einem Therapeuten zu machen. Oftmals ist der Hemmungsgrad grösser, wenn jemand zuschaut und der Patient vielleicht scheitert. Für dieses Problem eine Lösung zu finden ist die Aufgabe dieser Bachelorarbeit.

Die von Microsoft entwickelte Kamera, Kinect, kann als Eingabegerät für Xbox Spiele verwendet werden. Im Rahmen dieser Bachelorarbeit wird versucht zu eruieren, ob Kinect auch für Therapieübungen eingesetzt werden kann und wie eine solche Applikation aussehen sollte. Die verschiedenen Aspekte von Therapien wurden studiert. Sowohl für Hirnschlag-Patienten, welche Bewegungen neu erlernen müssen, als auch für Schwerbehinderte, welche sich kaum bewegen können und für die Kinect ein angenehmes Spielzeug ist. Im Laufe der Arbeit wurde zusätzlich die Einsatzmöglichkeit für Personen, welche bei der täglichen Arbeit viel sitzen, geprüft. Das heisst, ob auch Übungen zur Entspannung erstellt werden können.

Während der Einarbeitung in die Aufgabenstellung wurde untersucht, welche Applikationen und Fortschritte es in diesem Anwendungsgebiet von Kinect schon gibt und welche Module vielleicht für diese Arbeit weiter verwendet werden könnten. Es wurde versucht herauszufinden, was mit der Kinect möglich ist und wo die Grenzen liegen, dazu die beiden Kapitel Grundlagen und Konzeptionelle Überlegungen. Zusätzlich musste Kinect verstanden und die verschiedenen Funktionen kennengelernt werden.

In einem weiteren Schritt ist ein Software-Prototyp erstellt worden, in welchem die grundlegenden Funktionen der Applikation enthalten sind. Durch zusätzliche Erkenntnisse, welche durch Usability Tests mit Personen erhalten wurden, konnte die Applikation weiter den Bedürfnissen von potentiellen Benutzern angepasst und erweitert werden. Diese Ergebnisse sind im Kapitel Umsetzung und Usability-Tests dokumentiert.

1.1 Recherche

Die nachfolgenden Abschnitte beinhalten die Ergebnisse und Erkenntnisse aus der durchgeführten Recherche um das Thema «Einsatz der Kinect im Bereich der Medizin». In einem ersten Schritt wurde vor allem geprüft, welche Einsatzgebiete Kinect im medizinischen Bereich bereits hat und welche Ideen bereits umgesetzt worden sind. Zu einem späteren Zeitpunkt wurden konkrete Informationen zur Gestenerkennung ermittelt.

1.1.1 Ergebnisse «Einsatzgebiete im medizinischen Bereich»

Für diese Recherche wurde vorwiegend das Internet verwendet, weil abgesehen von wenigen Arbeiten bisher kaum brauchbare Literatur über dieses Thema erschienen ist.

Sehr schnell hat sich herausgestellt, dass Kinect vor allem in drei Bereichen eingesetzt wird:

- Bereits bestehende Kinect-Spiele, die in Behindertenheimen oder auch in Therapiezentren eingesetzt werden
- In Operationssälen, in welchen Kinect als Ersatz für die Maus verwendet wird
- Eigens für Therapiezwecke entwickelte Applikationen

Das letzte Gebiet ist für diese Bachelorarbeit das Interessanteste. Es soll eine Applikation entwickelt werden, welche den Menschen hilft, selbstständig Therapieübungen mit Rückmeldungen über die Korrektheit durchzuführen.

Das Ziel der speziell entwickelten Applikationen ist nicht nur die Kosteneinsparung durch Verzicht auf Übungsminuten mit dem Therapeuten, sondern auch, dass sich ein Patient nicht unter Druck und Beobachtung durch eine andere Person fühlt. Vielmehr soll er unbelastet selbstständig aufgrund der Rückmeldung durch das System die Übungen kontinuierlich erarbeiten und damit den Bewegungsablauf verbessern. Dieser Vorteil wurde insbesondere im Bericht von Ken Terry, www.informationweek.com, zum Naval Medical Center in San Diego hervorgehoben, welches Kinect für die Therapie von verwundeten Soldaten einsetzt. [8]

Generell hat sich herausgestellt, dass bisher in der Bewegungstherapie mit Kinect sehr gute Erfahrungen gemacht wurden. Im Bereich der Schlaganfallpatienten wurden durch den Einsatz von Spielkonsolen in der Therapie teilweise viel höhere Erfolgsquoten erreicht als ohne Kinect. Wie Chip Online in einem Artikel berichtet, konnten vor allem die motorischen Fähigkeiten deutlich verbessert werden. [3] Kinect wird aber auch bei älteren Menschen eingesetzt, welche feinmotorische Schwierigkeiten haben. Die polytechnische Universität Hong Kong hat in diesem Einsatzbereich bereits Erfolge verzeichnet. [21] Ebenfalls wurden autistische Kinder bereits Ziel von Kinect-Anwendungen. Die Behindertenhilfe Offenbach verweist dabei auf einen Artikel der Frankfurter Rundschau. [2] Gemäss diesem sollen Kinder mit der Kinect-Anwendung alleine lernen zu sprechen und ihre Aufmerksamkeit verbessern. Weiter gibt es Applikationen, bei welchen die Kinder lernen sollen, beide Körperhälften gleichzeitig einzusetzen. Auf der Homepage des Lakeside Center for Autism

wird die Technologie aufgezeigt. [12] Es gibt auch ein komplexeres System, welche mit Hilfe von Kinect den Gang von Personen therapieren. Die Webseite von Kinectotherapy zeigt Möglichkeiten des Systems auf. [11]

1.1.2 Ergebnisse «Gestenerkennung»

Prof. Dr. Josef M. Joller hat bereits in einer vergangenen Masterarbeit Studenten betreut, welche Kinect zur Gestenerkennung benutzt haben. Die entsprechende Applikation sowie die Dokumentation wurden zu Verfügung gestellt. Durch eine zusätzliche Recherche wurde versucht herauszufinden, ob Teile dieser Masterarbeit für diese Bachelorarbeit verwendet werden können. Die beiden Autoren, Philippe Morier und Martin Weber, haben in ihrer Arbeit «3D-Gesture Recognition» die Applikation und deren Entwicklung genauer erläutert. [18]

Weiter zeigen sie im Bericht zum Vorprojekt «Gestik zur Computer-Interaktion» verschiedene mögliche Algorithmen der Gestenerkennung auf. [19] Algorithmen wie «One Dollar» ($1 , Unistroke Recognizer), Johan Thelin (Recognizing Mouse Gestures), Per Ola Kristensson (Continous gesture recognition) oder den Algorithmus von Arnoud Icard werden aufgezeigt. Letzterer wurde in der Applikation verwendet und von den Studenten wie folgt begründet:

> [...] Im Fall einer Zögerung des Benutzers beim Zeichnen einer Geste entstehen Bereiche mit vielen Abtastpunkten. Dies führt zu einer künstlichen Verlängerung des Input-Pfades. Da beim $1 Algorithmus die Anzahl der Neuabtastungen von der Länge des Input-Pfades abhängt, entstehen im Bereich der Zögerung viele neue Punkte. Diese führen zu einer Verzerrung bei der Berechnung der durchschnittlichen euklidischen Distanz.[1] Beim Algorithmus von Arnoud Icard werden jedoch durch die Verwendung von Kreisen die unnötigen Abtastpunkte ignoriert. Dies führt allerdings zu einer anderen Anzahl Abtastpunkte im Vergleich zum Template. [...] [19]

Die im zitierten Text erwähnten zwei Phänomene, künstliche Verlängerung des Input-Pfades und die Algorithmus-abhängige Verzerrung, werden in den zwei nachstehenden Abbildungen 1.1 und 1.2 aufgezeigt. Die schwarze Linie in der Grafik zeigt den Input-Pfad des Benutzer. Der $1 Algorithmus und Arnoud Icard tasten diesen Pfad ab und finden die grün beziehungsweise blau markierten Punkte. Beide Algorithmen haben eine andere Anzahl Abtastpunkte, sodass es beim Vergleich mit der Vorlage zu einer Verzerrung kommt. Diese ist in der Abbildung 1.2 durch die orangen Linien gekennzeichnet.

[1] Die euklidische Distanz (der Abstand) zweier Punkte in der Ebene oder im Raum ist die zum Beispiel mit einem Lineal gemessene Länge einer Strecke, die diese zwei Punkte verbindet. [22] (Diese Begriffserklärung fehlt im originalen Text)

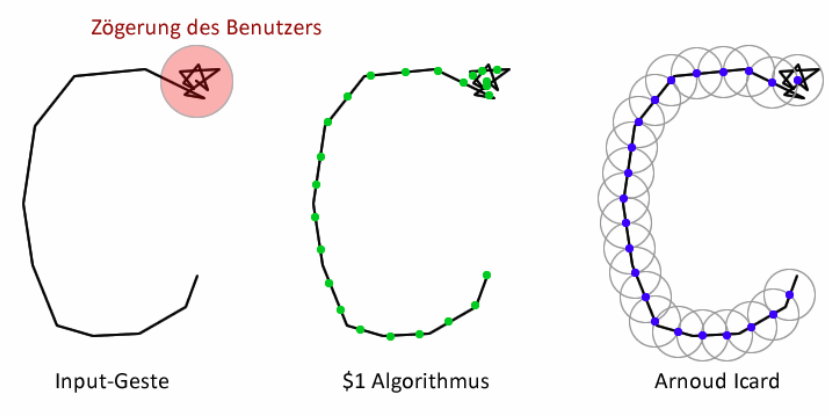


Abbildung 1.1: Problem der Neuabtastung bei $1 Algorithmus [18]

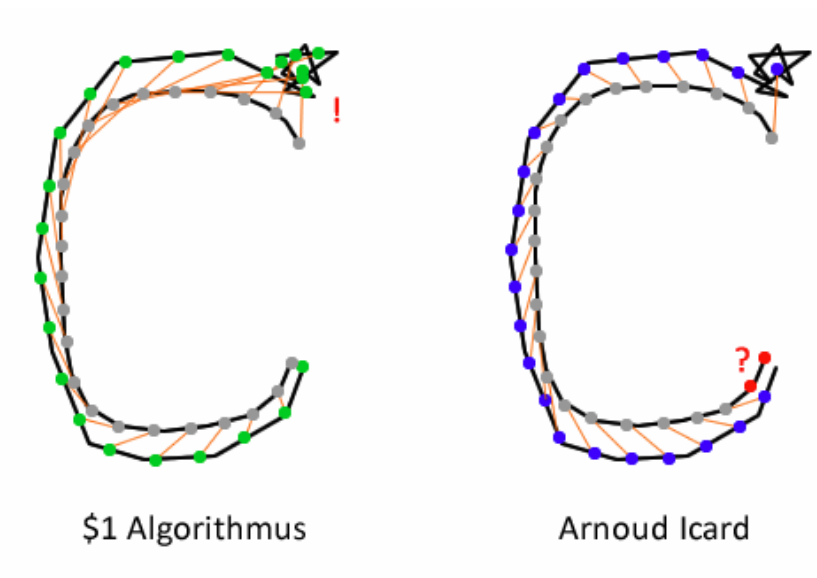


Abbildung 1.2: Algorithmus-abhängige Verzerrung [18]

Die entwickelte Applikation hat drei Modi: den «Benutzereingabe-Modus», den «Vorlage-Geste-Modus» sowie den «Betrachtungs-Modus». Diese drei Modi sind in der Abbildung 1.3 durch die Personen links gekennzeichnet. Letzterer lässt wie es der Name sagt, Gesten betrachten. Die anderen zwei Modi dienen zur Erstellung einer neuen Vorlage («Vorlage-Geste-Modus») beziehungsweise zur Benutzereingabe, in welcher eine Person eine Geste aufzeichnen kann und der Computer versucht diese einer Vorlage zu zuordnen («Benutzereingabe-Modus»). Als Referenz haben die Herren Morier und Weber die Koordinaten der rechten Hand verwendet.

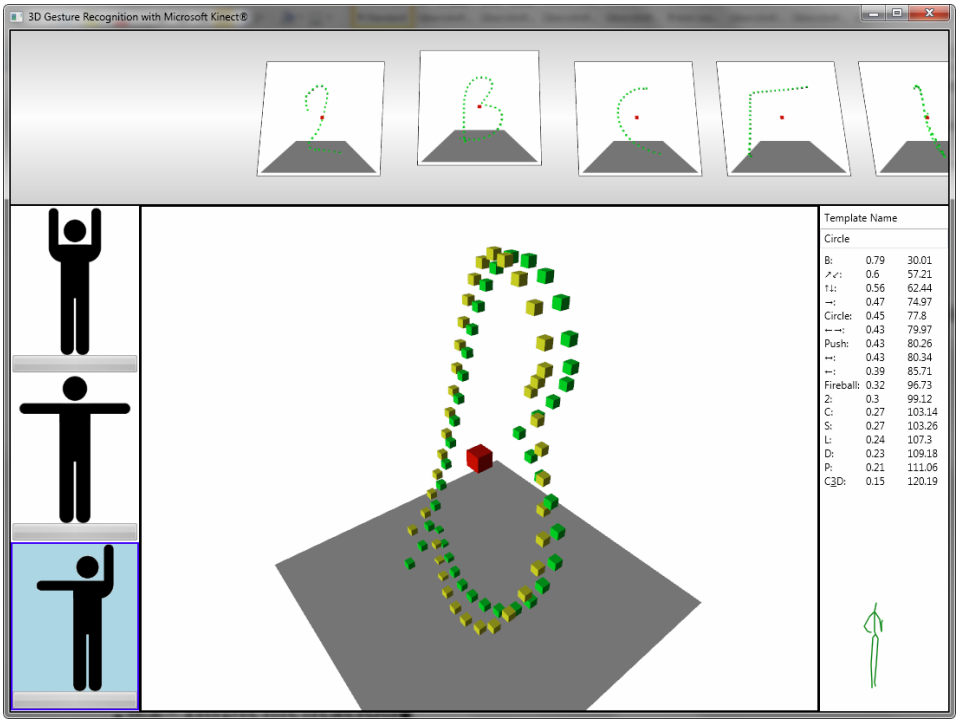


Abbildung 1.3: Screenshot: Demonstrationsapplikation [18]

Um eine Geste zu erfassen ist ein Start- beziehungsweise Stoppzeichen notwendig. Die beiden Studenten haben dies mittels einem Fussabstand von 20 cm in Richtung der Kinect-Kamera gelöst. Solange der eine Fuss 20 cm näher bei der Kinect ist, wird die Geste aufgenommen. Verringert sich die Distanz, so stoppt die Aufzeichnung. Wurde eine Gestenvorlage aufgenommen, kann diese mit einem entsprechenden Namen abgespeichert werden. Die Informationen über die Positionsangaben werden in einer XML-Datei gespeichert. So ist es möglich, gespeicherte Gesten zu einem späteren Zeitpunkt abzurufen (beispielsweise nach einem Neustart der Applikation).

Mit diesem Hintergrund wurde eine weitere Internetrecherche gestartet, um weitere Applikationen zur Gestenerkennung zu finden. Dabei konnte ein «Gesture Recording and Recognition Toolkit» namens «GesturePak for Kinect for Windows» gefunden werden. [5]

Dieses Toolkit bietet, ähnlich wie die obengenannte Masterarbeit, die Möglichkeit, Gesten aufzunehmen und abzuspeichern. Sobald eine Geste abgespeichert ist, kann diese auch wiedererkannt werden. Mit welchen Methoden und Algorithmen die Geste wiedererkannt wird, wird leider nicht beschrieben. Es ist davon auszugehen, dass die Erkennung auf bestehenden Algorithmen oder Kombinationen davon basiert.

Das Erfassen einer Geste geschieht voll umfänglich mit der Kinect. Die Bedienung erfolgt mit der rechten Hand. Diese imitiert dabei einen Mauszeiger. Bleibt die Hand eine längere Zeit am selben Ort, so wird ein Klick ausgelöst. Sämtliche Buttons sind jedoch auch in der Grammatik der Spracherkennung integriert und können mittels Spracheingabe angewählt werden. Jede Aufnahme einer Geste erlaubt es zudem genau zu definieren, welche

Gelenke und Bewegungsrichtungen betrachtet werden sollen. Auf der Abbildung 1.4 wird die Applikation vorgeführt. Es handelt sich dabei um einen Screenshot aus dem offiziellen YouTube Demonstrationsvideo.

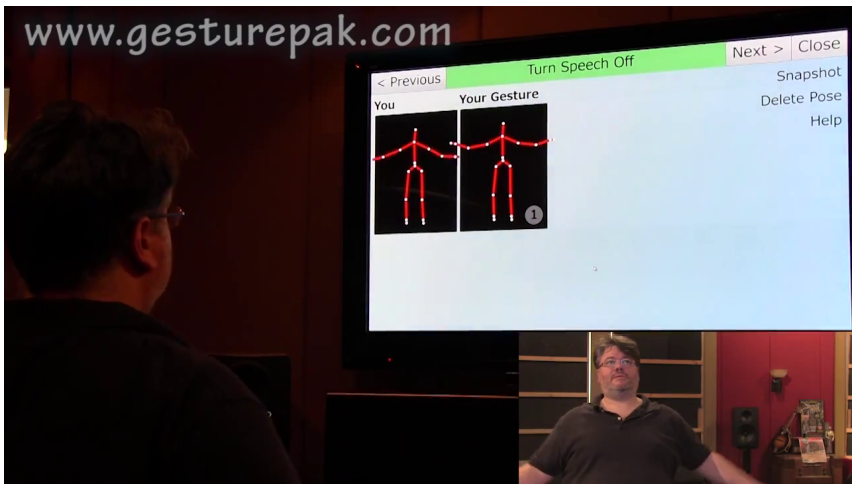


Abbildung 1.4: Screenshot: YouTube Demonstrationsvideo [25]

In einem Selbstversuch wurde die interaktive Demo, GesturePak, nachgespielt. Es handelt sich dabei um eine geführte Anleitung zum Kennenlernen der Applikation. Schlussendlich besteht auch die Möglichkeit, die Applikation selbstständig zu nutzen. Die gewonnen Erfahrungen haben gezeigt, dass die Anwendung stabil läuft und viele Möglichkeiten bietet. Lediglich die Spracherkennung hat nicht immer zu 100% funktioniert. Aus Sicht von Personen mit Behinderungen könnte dies zu einem grösseren Problem werden.

1.1.3 Fazit Recherche

Aus den Recherchen im Internet wurde erkannt, dass es bereits Therapie Applikationen mit Kinect gibt und diese auch erfolgreich benutzt werden. Es gibt jedoch in diesem Gebiet wie im folgenden angesprochen noch Potential zur Weiterentwicklung.

Die in der Masterarbeit von Philippe Morier und Martin Weber aufgezeigte Idee mit dem Steuern der Aufnahmefunktion durch den Schritt nach vorne kann in dieser Bachelorarbeit nicht eingesetzt werden. Sie ist für Personen mit Bewegungseinschränkungen als Start- und Stoppzeichen ungeeignet. Ein alternatives Zeichen wäre von Vorteil. Deshalb wurde im Kapitel Konzeptionelle Überlegungen versucht eine andere Lösung zu finden. Zusätzlich musste erkannt werden, dass die von ihnen in der Arbeit benutzten Algorithmen sich für diese Bachelorarbeit überlegte Idee nicht eignen. Die Übungen sollten sich nicht auf bestimmte, quasi statische, Figuren beschränken, sondern eher auf einfache Bewegungsabläufe mit dem Körper.

Die Analyse der Arbeit von Philippe Morier und Martin Weber hat diese Bachelorarbeit aber insofern voran gebracht als klarer wurde, was das exakte Ziel der Arbeit sein soll und wie dieses am Besten erreicht wird, um für das Zielpublikum interessant zu sein.

Auch aus der GesturePak-Applikation konnte die wichtige Erkenntnis gewonnen werden, dass die Möglichkeit der Sprachsteuerung eine Bedienung der Anwendung erlaubt. Inwiefern dies für das Zielpublikum der Bachelorarbeit geeignet ist, muss genauer betrachtet werden.

All diese Erkenntnisse waren sehr wichtig für das weitere Vorgehen in der Bachelorarbeit und sind bei der Entwicklung des Prototypen eingeflossen.

Kapitel 2

Grundlagen

In diesem Kapitel werden die Grundlagen, welche für das Verständnis der Bachelorarbeit notwendig sind, beschrieben und erklärt. Neben dem Kinect-Sensor wird das Software Developement Kit (kurz SDK) zu Kinect erläutert. Zusätzlich werden die Details zum Erkennen von Personen beschrieben und an einem Programmierbeispiel gezeigt.

2.1 Kinect-Sensor

Der Kinect-Sensor wurde 2010 von Microsoft für die Xbox 360 entwickelt. Mit der Kinect besteht die Möglichkeit, ein Spiel statt über ein Gamepad mit der Gestik des ganzen Körpers zu steuern. Microsoft erkannte die Notwendigkeit eines SDK und veröffentlichte dieses 2011, um die Entwicklung neuer Anwendungen für Windows 7 zu ermöglichen. Zusätzlich entwickelte Microsoft «Kinect for Windows», welche im Vergleich zur herkömmlichen Kinect für die Xbox, eine verbesserte Personenerkennung besitzt. Diese erschien ein Jahr später.

Nach einigen Versuchen mit beiden Kinect-Varianten wurde erkannt, dass für diese Bachelorarbeit die «Kinect for Xbox» verwendet werden kann. Deren Genauigkeit reicht bei der Personenerkennung für die erforderlichen Funktionen aus.

Nachfolgend ist der Kinect-Sensor der Kinect für Xbox beschrieben. Er besteht aus verschiedenen einzelnen Sensoren wie in der Abbildung 2.1 dargestellt.

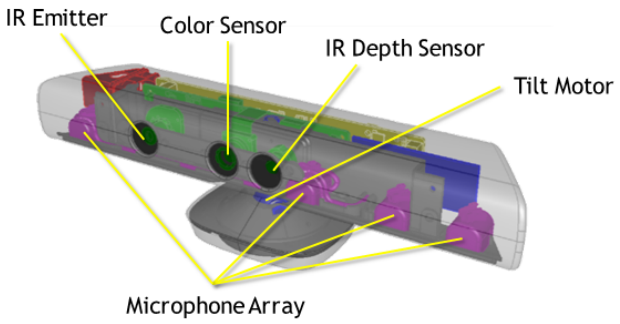


Abbildung 2.1: Komponenten des Kinect-Sensor [15]

Der Kinect-Sensor von Microsoft besteht aus den folgenden Komponenten:

- Einem Infrarot-Strahler («IR Emitter»). Dieser sendet Infrarotstrahlen in den Raum, die von den Messobjekten reflektiert werden und so eine Abstandsmessung zwischen Messobjekt und Sensor erlauben.
- Einem RGB-Sensor («Color Sensor»), welcher Farbbilder liefert. Je nach Auflösung unterscheiden sich die Anzahl der Frames pro Sekunde (FPS). 1280x960 Pixel mit 12 FPS oder 640x480 Pixel mit 30 FPS.
- Einem Infrarot-Tiefensensor («IR Depth Sensor»), welcher die Infrarotstrahlen detektiert und so den Abstand (Tiefe) der Messpunkte gegenüber des IR-Tiefensensors berechnet. Die Auflösung des IR-Sensors beträgt 640x480 Pixel bei einer Abtastrate von 30 Bildern pro Sekunde. Die Genauigkeit liegt bei ca. einem Zentimeter.
- Einem Kippmotor («Tilt Motor»), durch den der Kinect-Sensor um 27° nach oben und unten geneigt werden kann.
- Vier Mikrofone («Microphone Array»), durch welche das Geräusch oder der Befehl und Standort erkannt werden können.

2.2 Kinect-Software Developement Kit

Über das SDK ist es möglich, die verschiedenen Sensoren in einer Windows-Anwendung einzubinden. Für dieses Projekt wird das SDK in Version 1.7 (v1.7) verwendet. Das SDK bietet unter anderem die Möglichkeit, ein Skelett zu tracken. Das heisst, die Körperteile einer Person können erkannt und nachverfolgt werden. In dem SDK v1.7 sind zusätzliche Erweiterungen enthalten, welche viele Vorteile für die Bachelorarbeit bieten.

Eine davon ist die sogenannte «Kinect Interaction». Dieses Feature bietet die Funktionen «push» und «grip». «Push» ermöglicht es virtuelle Objekte zu selektieren, wie beispielsweise das Klicken von Buttons. Die Gestik dafür ist sehr einfach und intuitiv. Man stösst die Hand nach vorne, als wolle man tatsächlich einen Button anklicken. «Grip» hingegen bietet die Möglichkeit auf einer Seite zu scrollen. Dies wird gemacht, indem die Hand zur Faust geschlossen wird, was als Zupacken simuliert wird.

2.2.1 Skeleton-Tracking

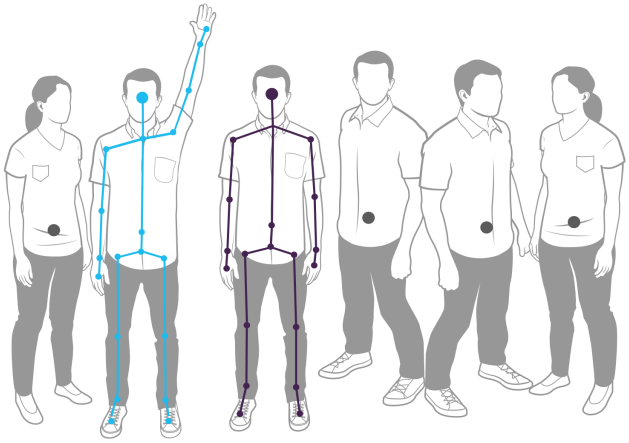


Abbildung 2.2: von der Kinect erkannte und getrackte Personen [4]

Wie in der Abbildung 2.2 dargestellt, kann die Kinect-Software maximal sechs Personen gleichzeitig erkennen. Davon aber nur zwei Personen tracken, wobei sie pro Person ein «Skeleton» (zu Deutsch Skelett) erkennt. Dieses umfasst 20 Punkte am Körper, die als «Joints» repräsentiert werden. Von dem SDK werden die in der Abbildung 2.3 beschriebenen Joints (zu Deutsch «Gelenke») erkannt. Diese Gelenke wiederum besitzen drei Werte, welche den kartesischen Koordinaten des Gelenkpunkts im X, Y und Z Koordinatensystem von Kinect entsprechen. Diese Informationen werden in einem «SkeletonPoint» gespeichert. Damit eine Person erkannt wird, sprich diese «SkeletonPoints» von der Software erkannt werden können, muss der Kopf sowie der Rumpf dem Kinect-Sensor zugewandt sein.

Die Möglichkeit, diese zwanzig Punkte und somit der ganze Körper einer Person zu erkennen, ist erst seit kurzem möglich. Vorher gab es eine einfachere Version mit nur zehn Punkten (Kopf, Arme und Hals).

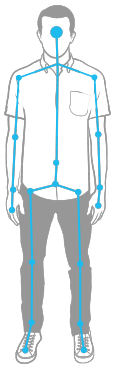


Abbildung 2.3: Person charakterisiert mit 20 «SkeletonPoints» [4]

2.2.2 Programmierbeispiel

In den nachfolgenden Codezeilen, wird beschrieben, wie ein «Skeleton» mit der Kinect getracked werden kann. [6] [16]

Dafür werden die folgenden Variablen benötigt:

```
1 Skeleton[] skeletonData;
2 Skeleton skeleton;
```

Listing 2.1: Initialisierung der Variablen

In der Variable «skeleton» wird das aufgenommene Skelett gesichert. In «skeletonData» werden alle erkannten Skelette gespeichert. Danach muss der «SkeletonStream» eingeschaltet werden, damit die Kinect-Software das Skelett zu erkennen beginnt.

```
1 kinect.SkeletonStream.Enable();
```

Listing 2.2: «SkeletonStream» aktivieren

Die Variable «kinect» repräsentiert den Kinect-Sensor. Als nächstes muss man eine Methode schreiben, welche jedes mal, wenn ein Frame von der Kinect-Software empfangen wird, ausgeführt wird.

```
1 kinect.SkeletonFrameReady += new EventHandler<
      SkeletonFrameReadyEventArgs>(kinect_SkeletonFrameReady);
```

Listing 2.3: EventHandler hinzufügen

Die Methode öffnet ein SkeletonFrame. Falls dieses vorhanden ist, werden die Skeleton-Daten in die vorher angelegte Variable «skeletonFrame» geschrieben.

```
1  private void kinect_SkeletonFrameReady(object sender,
       SkeletonFrameReadyEventArgs e)
2  {
3      using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
4      {
5          if (skeletonFrame != null && this.skeletonData != null)
6          {
7              skeletonFrame.CopySkeletonDataTo(this.skeletonData)
                   ;
8          }
9      }
10 }
```

Listing 2.4: «SkeletonFrameReady»-Methode

Weiter muss das Skelett gezeichnet werden. Dazu wird durch die «skeletonData» iteriert und für jedes Skelett herausgefunden, in welchem Status es ist. Ein «Skeleton» kann den Status (SkeletonTrackingState) «Tracked», «PositionOnly» oder «NonTracked» haben. «Tracked» bedeutet, dass alle 20 Gelenkpositionen erkannt wurden. Beim «PositionOnly»-Status ist nur der Ort des ganzen Skeletts bekannt, sprich wie oben erwähnt, können zwei Personen den Status «Tracked» und vier den Status «PositionOnly» haben.

```
1  private void DrawSkeletons()
2  {
3      foreach (Skeleton skeleton in this.skeletonData)
4      {
5          if (skeleton.TrackingState == SkeletonTrackingState.
               Tracked)
6          {
7              // Zeichnet jeden erkannten Joint als Punkte
8              DrawTrackedSkeletonJoints(skeleton.Joints);
9          }
10         else if (skeleton.TrackingState ==
               SkeletonTrackingState.PositionOnly)
11         {
12             // Zeichnet die Position der Person
13             DrawSkeletonPosition(skeleton.Position);
14         }
15     }
16 }
```

Listing 2.5: «DrawSkeletons»-Methode

Diese Code-Beispiele fungierten als Grundlage für die weiteren Entwicklungen.

Kapitel 3

Konzeptionelle Überlegungen

Dieses Kapitel handelt von Konzepten, die im Laufe der Bachelorarbeit erarbeitet wurden. Es zeigt auf, welche Überlegungen dazu gemacht wurden und welche Entscheidungen getroffen worden sind.

3.1 Startzeichen für eine Übungsaufzeichnung

In der Masterarbeit der Herren Morier und Weber kann eine Bewegung aufgenommen werden. Hierfür muss ein Fuss einen Schritt nach vorne gesetzt werden. Dabei stellt sich die Frage, ob es keine andere Möglichkeit gibt. Ein Patient, welcher eine Einschränkung bei der Beweglichkeit der Beine hat, hat keine Chance eine Übung aufzunehmen und diese auszuführen. Verschiedene Möglichkeiten wurden untersucht und die Vor- sowie Nachteile eruiert. Jeder Patient soll die Möglichkeit haben, eine Übung auszuführen.

3.1.1 Face-Tracking

In einem Versuch wurde die Gesichtserkennung, welche in einem Anwendungsbeispiel von Kinect verwendet wird, ausprobiert. Ziel war es herauszufinden, ob ein geöffneter Mund als mögliches Start oder Stoppzeichen verwendet werden kann. Konkret wäre das eine Alternative für die Fussstellungsdifferenz, welche beim «GestureRecognizer» notwendig ist. Das Gesicht wird dabei in viele kleinere und grössere Dreiecke («Triangle») aufgeteilt. Um die Programmierung den Entwicklern zu vereinfachen, wird dies im SDK zu Verfügung gestellt. Dieses beinhaltet unter anderem sechs sogenannte «Animation-Units» [14]:

- AU0 – Upper Lip Raiser
  Unterscheidung zwischen neutralem Gesichtsausdruck (Zähne durch die Lippe verdeckt), geöffnete (Zähne sind voll umfänglich sichtbar) und geschlossene Lippen (Lippen sind zusammen gepresst).
- AU1 – Jaw Lowerer
  Unterscheidung zwischen geschlossenem und geöffneten Kiefer.

- AU2 – Lip Stretcher
  Unterscheidung zwischen neutralem Gesichtsausdruck, «gestreckten» Lippen (sogenanntes «Joker's smile»), einem Schmollmund und «gerundeten» Lippen («Kissing mouth»).

- AU3 – Brow Lowerer
  Unterscheidung zwischen neutralen Augenbrauenstellung, völlig gesenkten und gehobenen Augenbrauen.

- AU4 – Lip Corner Depressor
  Unterscheidung zwischen neutraler Mundwinkelstellung, einem glücklichen Lächeln und einem traurigen Gesichtsausdruck.

- AU5 – Outer Brow Raiser
  Unterscheidung zwischen neutralen, gesenkten (trauriger Gesichtsausdruck) und gehobenen Augenbrauen (überraschter Gesichtsausdruck).

Diese Animation-Units erlauben dem Entwickler, Gesichtsausdrücke einfacher zu erkennen. Es können lediglich die Werte abgefragt und ausgewertet werden.

Um den Versuch zu starten wurde als Vorlage das Beispiel («Face Tracking Basics-WPF») von Microsoft verwendet. Dieses verwendet das FaceTracking SDK und zeigt die Dreiecke bereits auf. Der Code kann ergänzt werden, indem der «AU1» abgefragt wird, um durch einen geöffneten Mund die Aufnahme zu starten. Der Wert zeigt auf, ob der Kiefer (und somit das Mund) geöffnet oder geschlossen ist. In der Abbildung 3.1 wird dieser links oben angezeigt. Der Kiefer war während der Aufnahme geöffnet, sodass ein positiver «AU1»-Wert resultiert.

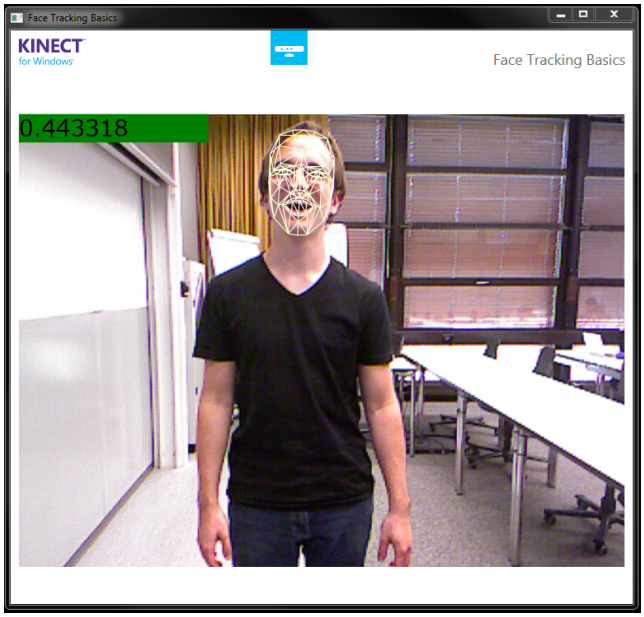


Abbildung 3.1: Screenshot: «FaceTracking» mit Kinect inklusive Anzeige des «AU1»

Schnell hat sich gezeigt, dass die Berechnung stark schwanken kann und die Gestik nicht zu hundert Prozent richtig erkannt wird. Weiter kommt hinzu, dass Kinect für die Gesichtserkennung für eine kurze Distanz zwischen Kinect-Sensor und dem Gesicht gedacht ist. Je weiter weg die Person ist, desto weniger Pixel stehen für das Gesicht zu Verfügung. Da zudem im Rahmen der Bachelorarbeit der ganze Körper für Übungen verwendet werden soll, ist die Distanz für die Gesichtserkennung eher zu gross.

Die Idee der Gesichtserkennung scheint zwar spannend, je nach Einsatzbereich ist doch kein praktikabler Einsatz möglich.

3.1.2 Spracherkennung mit Kinect

Der Kinect-Sensor hat gemäss den technischen Spezifikationen vier Mikrofone eingebaut, welche zur Spracherkennung verwendet werden können. Eine Grammatik mit Wörtern zur Erkennung kann vom Entwickler festgelegt werden. In einem Versuch wurde analysiert, inwiefern dies bei dieser Bachelorarbeit für Befehle verwendet werden könnte. Gefragt waren insbesondere Trefferquote beim Erkennen, Störungen durch das «Umfeld» sowie das Befinden einer Person.

In einem ersten Versuch wurde eine Grammatik mit den drei Begriffen: «New», «Track» und «Calculate» definiert. Jeder Begriff hat eine andere Methode in der Applikation aufgerufen. Eine Testperson kann einen dieser Begriffe nennen und Kinect versucht, den Begriff zu erkennen (siehe Abbildung 3.2). Wird dieser erkannt, wird die angegebene Methode ausgeführt.

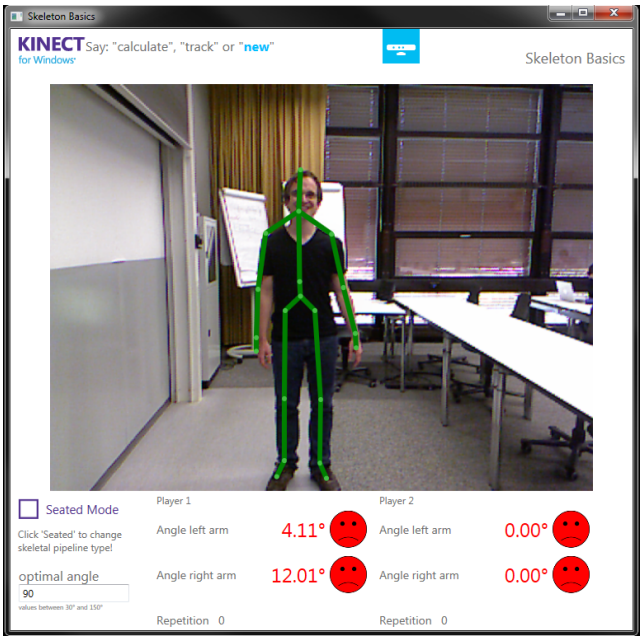


Abbildung 3.2: Screenshot: Spracherkennung, Begriff «new» wurde erkannt

Der erste Versuch fand im Zimmer der Bachelorarbeiten statt. Der Raum war besetzt mit anderen Mitstudenten, welche sich zum Teil unterhielten. Der allgemeine Geräuschpegel war jedoch nicht sehr laut. Die Applikation wurde gestartet und der erste Begriff genannt. Kinect hat diesen jedoch erst nach mehrmaligem Wiederholen erkannt. Ebenso wurden Begriffe erkannt, welche nicht gesagt wurden. Es muss sich dabei wohl eher um Fehlinterpretationen von der Kinect gehandelt haben.

Generell kann die Trefferquote der Spracherkennung als genügend eingestuft werden. Störungen durch das Umfeld können zu Fehlinterpretationen führen. Durch die nicht 100%ige Erkennung der Sprache und Fehlinterpretationen ist es für den Benutzer eher irreführend und macht nicht wirklich Freude. Selbstverständlich müssten für eine konkrete Angabe weitere Tests gemacht werden. Zum Beispiel könnten die Parameter der Erkennung verändert werden. So könnte der Wert, um wie viel Prozent sich die Applikation sicher ist, dass das Wort übereinstimmt, verändert werden. Allerdings ist davon auszugehen, dass die Spracherkennung für das Einsatzgebiet dieser Bachelorarbeit nicht geeignet ist. Patienten mit einer Sprachbehinderung sollen ebenfalls Übungen durchführen können.

3.1.3 Entscheidung

Im Rahmen dieser Bachelorarbeit ist die Gesichtserkennung durch die oben genannten Gründe nicht möglich. Die Spracherkennung im Moment noch zu ungenau und das erfolgreiche Aufnehmen hängt von diversen Faktoren ab. So wurden diese beiden Varianten schnell verworfen. Da die Überlegung von geeigneten Übungsausführungen noch nicht weiter besprochen wurden, wurde entschieden, die Methodik zur Aufnahme der Bewegungen vorläufig beiseite zu legen. Falls die Ausführung der Übung dies noch als notwendig erachtet, kann später nach einer geeigneten Möglichkeit gesucht werden.

3.2 Übungsdurchführung

Schnell kamen neue Fragen auf:

- Wie können die Übungen am Besten von der Kinect erkannt werden?
- Wie können diese überprüft werden?
- Welche Möglichkeiten bietet die Kinect dafür?

Die nachfolgenden Abschnitte versuchen diese Fragen zu beantworten. Dabei wurde das Zielpublikum der Applikation nicht vorgegeben. Es wurde deshalb kein Augenmerk auf die Bewegungseinschränkung der Patienten gelegt.

3.2.1 Winkelberechnung der Gelenkstellungen

Im Internet gibt es bereits eine Lösung, die KinectoTherapy [11], welche sich auf die Winkel der Gelenke konzentriert. Sie bieten die Möglichkeit an, den Winkel der Arme und Beine zu messen. Dies wurde in einem ersten Versuch nachgebaut.

Dabei zeigte sich schnell, dass der Einsatz von Kinect nicht ganz trivial ist. Für die Berechnung eines Armwinkels sind geeignete Positionsangaben notwendig. Zuerst wurden dazu die Werte der Hüfte rechts, der Schulter rechts und des rechten Ellbogens gewählt. Aufgrund dieser drei Positionen wurde der Winkel des Arms bei der Schulter berechnet. Falls jedoch der Benutzer in den «Seated Mode» wechselt, verliert man die Angabe der Hüfte. Der «Seated Mode» wird verwendet, um eine sitzende Person zu erkennen. Um diese Möglichkeit offen zu lassen, wurde die Koordinate des Kopfes genommen. Diese ist sowohl für den normalen Modus, wie auch den «Seated Mode» verfügbar.

Um die Berechnung etwas zu vereinfachen, wurde dabei eine kleine Anpassung vorgenommen. Für die Winkelberechnung wird die X-Position des Kopfes auf die X-Position der Schulter gelegt (siehe neuen Position K‘ auf der Abbildung 3.3). Dadurch entsteht eine vertikale Linie (b) vom verschobenen Kopf zur Schulter.

Bei den ersten Versuchen mit dieser Änderung fiel weiter auf, dass die Berechnung nicht in jedem Fall stimmt. Wird beispielsweise nur ein Arm (der rechte) hochgehoben, so besteht die Möglichkeit, dass der Kopf nach links geneigt wird. Kinect verschiebt aufgrund dieser Neigung die Positionen der Schulter sowie alle Positionen des rechten Arms. Die Berechnung stimmt dann nicht vollständig. Ebenfalls muss für die Berechnung des Winkels, die Position der Hand benutzt werden. Der Ellbogen befindet sich, aufgrund der physikalischen Einschränkung, nicht auf der gleichen Y-Höhe wie die Schulter und die Handfläche.

Unter Berücksichtigung dieser Erkenntnisse kann die Winkelberechnung gemäss nachstehender Skizze (Abbildung 3.3) ausgeführt werden. Die Tabelle 3.1 zeigt die Legende zur Grafik.

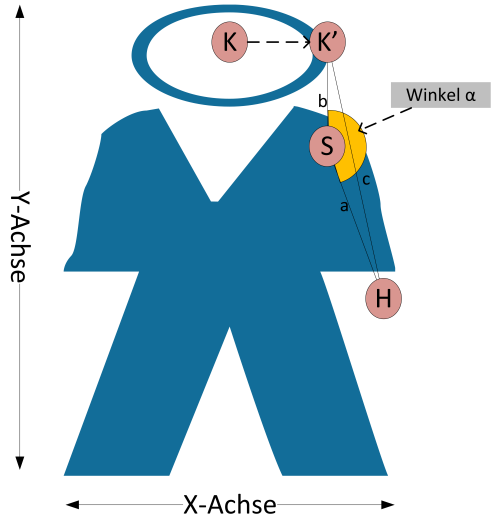


Abbildung 3.3: Skizze der Winkelberechnung

Symbol	**Beschreibung**
K	Kennzeichnet den Punkt, den Kinect als Kopfposition auswertet.
K'	Kennzeichnet den Punkt, der aus der Kopfposition und der Schulter abgeleitet wurde. Die Y-Position entspricht der Y-Position des Kopfes. Die X-Position entspricht die der rechten Schulter. Kurz gesagt wurde der Kopf-Punkt nach rechts verschoben.
S	Kennzeichnet den Punkt, den Kinect als Position der Schulter rechts findet.
H	Kennzeichnet den Punkt, den Kinect als Position der Hand rechts findet.
a (Linie)	Kennzeichnet die Linie zwischen Schulter und Hand.
b (Linie)	Kennzeichnet die Linie zwischen Schulter und Kopf.
c (Linie)	Kennzeichnet die Linie zwischen Kopf und Hand.
α (Winkel)	Kennzeichnet den «oberen» Winkel, den die Hand generiert im Vergleich zum Kopf.

Tabelle 3.1: Legende zu «Skizze der Winkelberechnung»

Die Formel für den Winkel α lautet wie folgt: $\alpha = \arccos(\frac{a^2+b^2-c^2}{2ab})$. Das berechnete α entspricht nun dem Winkel im Bogenmass. Um den Winkel in Grad zu erhalten, muss der Wert im Bogenmass umgerechnet werden. Weiter muss der Wert von 180° abgezogen werden, da die Berechnung von «Oben» getätigt wird. Unter Berücksichtigung dieser Anpassungen resultiert die Formel: $\gamma = 180 - (\frac{\alpha * 180}{\pi}) = 180 - (\frac{\arccos(\frac{a^2+b^2-c^2}{2ab}) * 180}{\pi})$. Diese Berechnung funktioniert nun in beiden Modi.

Auf der Abbildung 3.4 wird ein Screenshot des ergänzten Beispiels «Skeleton Basics» von Microsoft gezeigt, welches um die Winkelberechnung erweitert wurde.

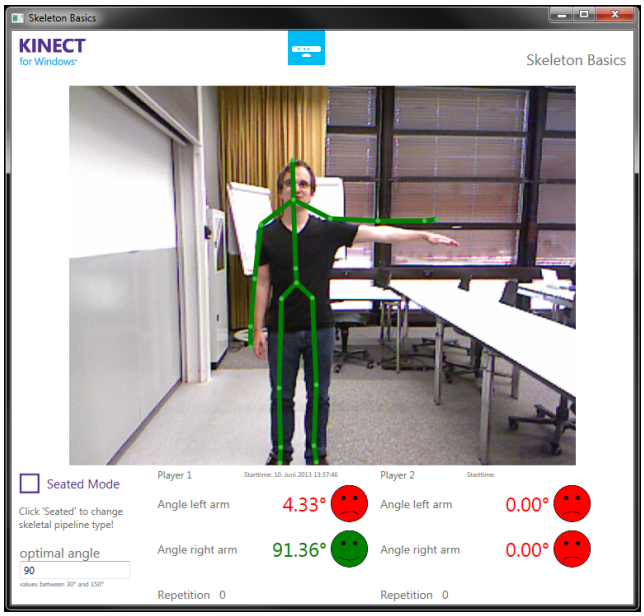


Abbildung 3.4: Screenshot: Beispiel «Skeleton Basics» von Microsoft ergänzt mit der Winkelberechnung

Das Buch «Checkliste Orthopädie» von Andreas B. Imhoff, Ralf D. Linke und René Baumgartner liefert Hinweise zu den medizinischen Bewegungsmöglichkeiten der Gelenke und Wirbelsäule. [1] Gemäss diesem scheint beim Hochheben des Armes ein Winkel von 90° optimal. Bei den Beinen kann ein Winkel von 40° als optimal angesehen werden. Viel mehr ist mit einem Bein nicht möglich. Die genauen Werte sollten jedoch mit einem Therapeuten näher betrachtet und validiert werden.

Wie sich in einem weiteren Versuch gezeigt hat, ist die Berechnung des Winkels noch mit weiteren Hürden verbunden. Streckt eine Person den Arm nicht durch, hält diesen aber in die Höhe, wird trotzdem ein Winkel berechnet. Dieser entspricht nicht ganz der Realität, da der Arm nicht durchgestreckt ist. Dies kann abgefangen werden, indem die Y-Koordinaten der Schulter und der Hand gemittelt wird und die Abweichung der Y-Koordinate des Ellbogens betrachtet wird. Ist diese Abweichung zu gross, so sollte eine «Fehlermeldung» erscheinen, dass die Person den Armen ganz durchstreckt. Ebenso sollte diese Meldung kommen, wenn eine Person den Ellbogen nach hinten gedreht hat und so versucht das System zu hintergehen.

Wie nun die Bewegungsübung im User Interface angezeigt werden soll, konnte noch nicht geklärt werden. Zum Beispiel stellt sich die Frage, ob das Skelett, welches die Kinect erkennt, oder die reale Person angezeigt werden soll oder beides. Diese Frage müsste in einem experimentellen Versuch weiter untersucht werden. Ebenso stellt sich die Frage, wie schnell sich die Anzeige anpassen soll. Geschieht dies zu schnell, kann dies von der Testperson als störend empfunden werden. Werden die Ausgaben «verzögert», aktualisie-

ren sich die Werte nicht gleich oft. Das kann auch wirken, als sei das Programm einfach viel zu langsam und mit der Berechnung nicht nachkommt.

Da in verschiedenen Diskussionen wiederholt zu Sprache kam, dass auch ein Mehrpersonen-Modus möglich sein sollte, müssen auch Personen unterschieden werden können. Steht Person A und Person B vor dem Kinect-Sensor so werden diese als Person 1 und Person 2 erfasst. Verlässt Person 1 den Erfassungsbereich von Kinect und kommt später zurück, so wird diese nicht mehr erkannt. Es kann keine zusätzliche Person mehr erfasst werden, weil der Platz noch mit der Ersten besetzt ist. Diese wird aber von Kinect nicht wieder als Person 1 erkannt. Dieses Problem kann jedoch so umgangen werden, dass wenn eine Person fünf Sekunden lang nicht mehr getracked werden kann, diese Person als von der Bildfläche verschwunden gilt. Die Person wird freigegeben.

3.2.2 Generische Position der Gelenke

Ein weiterer Ansatz um die Übungsdurchführung zu überprüfen, ist die Positionen der Gelenke zu erfassen und auf die Richtigkeit zu vergleichen. Dafür muss allerdings die Übung zuerst aufgenommen werden, damit ein Vergleichswert vorhanden ist. Für das Erstellen einer neuen Übung sollten sämtliche von Kinect gefundene Gelenke erfasst werden können.

Zuerst muss ein Weg gefunden werden, um die erfassten Werte so speichern zu können, damit zu einem späteren Zeitpunkt eine Übereinstimmung geprüft werden kann. Da sich jede Person leicht anders bewegt, macht es Sinn, lediglich Eckdaten zu erfassen und eine Übung in Teilschritte zu unterteilen. Stimmen diese Teilschritte überein, so gilt die Repetition als erfüllt. Je nach Übung sollten mehr oder weniger Teilschritte aufgezeichnet werden.

Die weitaus kompliziertere Aufgabe ist es, einen Weg zu finden, um eine Übereinstimmung zwischen der Vorgabe und den aktuellen Daten zu prüfen. Als kleines Beispiel: Eine Person will eine Übung erstellen. In dieser Übung hebt sie den Arm, damit ein Winkel von 90° entsteht. Werden dabei die absoluten Positionsangaben gespeichert, ist die Wahrscheinlichkeit, ein zweites Mal genau die selben Positionen zu erreichen, nahe 0. Hierfür müsste deshalb eine relative Position gewählt werden.

Die relative Position muss dabei so gewählt werden, dass sie allgemein gültig ist. Dies hat zur Idee geführt, eine Art «virtuellen Quader» um die Person zu bilden. Um einen solchen Quader zu bilden, benötigt es eine Kalibration. Die Person muss hierfür eine Grundposition einnehmen, bei welcher alle Gelenkpunkte zu sehen sind. Die Werte werden bei der Kalibration zusammengetragen und es wird eine Grundposition festgelegt, mit welcher der Quader gebildet werden kann. Die Höhe und die Breite kann mit Hilfe der Körpergrösse sowie der Arm- und Beinlänge bestimmt werden. Da die mögliche Distanz zur Kinect eingeschränkt ist, kann die Tiefe durch einen fixen Wert von einem Meter nach vorne und nach hinten bestimmt werden.

Mit Hilfe dieses Quaders lässt sich dann eine relative Position in X-, Y- und / oder Z-Richtung bestimmen. Um Ungleichheiten des Körperbaus vorzubeugen wurde bei der X- und Y-Richtung die Position der Wirbelsäule («Spine») als Mittelpunkt gewählt.

Für das Erfassen von neuen Übungen werden die relativen Positionen der einzelnen Gelenke gespeichert. Für die Übungsdurchführung werden die aktuellen relativen Positionen mit der Vorgabe verglichen.

3.2.3 Entscheidung

Die Winkelberechnung wäre insofern einfach zu realisieren gewesen, dass nicht zuerst eine Übung erstellt werden muss. Es ist lediglich eine Vorgabe notwendig und kann gleich ausgeführt werden. Allerdings ist die Einschränkung dieser Methode sehr gross. Nicht nur, dass nur eine beschränkte Anzahl verschiedener Winkel berechnet werden können, sondern auch, dass ein Therapeut die technischen Kenntnisse haben muss. Eine Übung müsste von ihm programmiert werden. Die Wahrscheinlichkeit, dass ein Therapeut eine Übung selbst erstellen beziehungsweise programmieren kann, ist gering, was diesen Ansatz als schwierig erachten lässt.

Im Vergleich dazu sind die Möglichkeiten, welche die Variante mit den generischen Positionen der Gelenke bietet, viel grösser. Eine Zweitperson kann eine Übung ohne weitere Kenntnisse erstellen und spezifische Körperteile kontrollieren. Der Patient kann diese einfach ausführen.

Durch die Einschränkung, welche die Winkelberechnung mit sich bringt, ist die Entscheidung sehr leicht gefallen, dass die generischen Übungen benutzt werden.

3.3 Aufzeichnung der Übungen

Für einen Benutzer ist es angenehm, wenn er gleich sieht, wie die Übung ablaufen soll. Um dies zu ermöglichen, wurde überlegt die Koordinaten einer Bewegung zwischen zu speichern und diese danach abzuspielen. Dies entspräche einem Video. Um diese Möglichkeit zu testen, wurde eine Beispiel Applikation geschrieben. Diese speichert die ausgeführte Bewegung und spielt sie danach verzögert ab. Dadurch, dass die Aufnahme gleich abgespielt wird, kann sofort erkannt werden, ob diese Möglichkeit funktioniert und die Punkte richtig dargestellt werden.

Für diesen Versuch wurde der Entwurf der Winkelberechnungen um ein weiteres «image»-Feld ergänzt. Das linke Bild zeigt dabei die aktuelle Aufnahme und das Rechte die verzögerte Bewegung (siehe Abbildung 3.5). Bei der Darstellung der aktuellen Aufnahme wird dabei jedes Skelett mit dem «SkeletonTrackingState» «Tracked» in eine Liste gespeichert. Diese wird gebraucht um die gespeicherten «Skeleton»'s abrufen zu können.

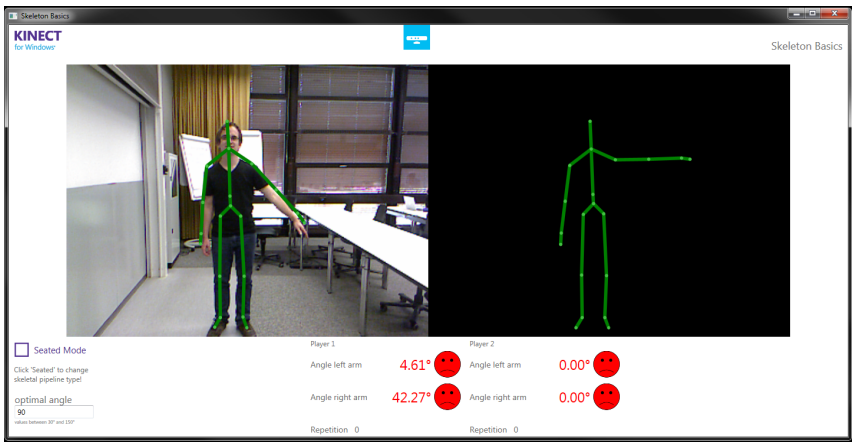


Abbildung 3.5: Screenshot: Verzögerte Darstellung

Der Abrufmechanismus kann dabei in die «SensorColorFrameReady»-Methode eingefügt werden. Diese wird bei jedem neuem Bild (dem «ColorFrame») aufgerufen und eignet sich somit am Besten für die verzögerte Darstellung. Sie hat die Aufgabe, das Bild auf der grafischen Oberfläche darzustellen. Sofern der Kinect-Sensor angeschlossen ist und die Applikation läuft, wird diese Methode immer aufgerufen. Anders sieht es bei «SensorSkeletonFrameReady» aus. Diese wird nur aufgerufen, wenn ein Skelett erkennt wird.

Für die verzögerte Darstellung wurden folgende Überlegungen gemacht:

- Das Skelett soll verzögert dargestellt werden. Also beispielsweise mit einer Verzögerung von 50 Millisekunden.
- Die Anzeige soll jeweils nur ein Skelett einblenden. Es soll sich dabei um das älteste Skeleton handeln.

Aufgrund dieser Überlegungen kann die Verzögerung implementiert werden und auf dem rechten «image»-Feld dargestellt werden. Dies klappt soweit ganz gut. Auch besteht

die Möglichkeit, eine grössere Verzögerung einzubauen und die Skelette auf Knopfdruck anzuzeigen. Die Verzögerung zwischen den einzelnen Aufnahmen kann selbstverständlich ebenfalls angepasst werden.

Die Speicherung des Skeletts in einer XML-Datei scheint jedoch für diese Bachelorarbeit nicht die beste Variante zu sein. Ein «Skeleton»-Objekt beinhaltet Werte für bis zu 20 Gelenke. Jedes Gelenk hat zusätzlich drei Werte mit den Angaben zur Koordinate. Aufgrund der Kinect-Konfiguration werden in einer Sekunde 30 «Skeleton»-Objekte erstellt. Die Datei würde sehr gross werden und das Einlesen, um die Aufnahme abzuspielen, dementsprechend lange gehen.

Ein weitere Ansatz ist ein Video aufzunehmen. Da die Aufnahme nur den Ablauf der Übung zeigen soll, ist es nicht notwendig die genauen Koordinate zu kennen. Für die Videoaufnahme wird während acht Sekunden jedes Frame, welches die Kinect über den «ColorStream» aufnimmt, als Bild in eine Liste gespeichert. Danach wird aus den Bildern ein Video erzeugt. Dies zeigen die untenstehenden Code Ausschnitte:

```
1  private void SaveFile()
2  {
3    int j = 0;
4    // Iteration durch die Liste der aufgenommene Bilder
5    foreach (WriteableBitmap item in _bitmapList)
6    {
7        _filename = j++ + ".png";
8        using (var stream = new FileStream(Path.Combine(
             MainWindow.MyNewExercise.ExercisePath, _filename),
             FileMode.Create))
9        {
10           var encoder = new PngBitmapEncoder();
11           encoder.Frames.Add(BitmapFrame.Create(item));
12           encoder.Save(stream);
13           stream.Close();
14       }
15   }
16 }
```

Listing 3.1: Methode «SaveFile»

```
1  private Boolean GenerateVideo()
2  {
3      string filename = Path.Combine(MainWindow.MyNewExercise.
           ExercisePath, "1.png");
4
5      MyNewExercise.VideoPath = String.Format(MyNewExercise.Date)
           + ".avi";
6
7      var bitmap = (Bitmap) Image.FromFile(filename);
8
9      var aviManager = new AviManager(Path.Combine(MyNewExercise.
           ExercisePath, MyNewExercise.VideoPath), false);
10
11     VideoStream aviStream = aviManager.AddVideoStream(false,
           25, bitmap);
```

```
12
13      for (int i = 1; i < MaxVideoPictures; i++)
14      {
15          if (File.Exists(Path.Combine(MyNewExercise.ExercisePath
                , i + ".png")))
16          {
17              var tempBitmap = (Bitmap) Image.FromFile(Path.
                    Combine(MyNewExercise.ExercisePath, i + ".png")
                    );
18              aviStream.AddFrame(tempBitmap);
19              bitmap.Dispose();
20          }
21          ProgressbarConvertingValue++;
22      }
23      aviManager.Close();
24      return true;
25  }
```

Listing 3.2: Methode «GenerateVideo»

Die Entscheidung, ob ein Video erzeugt oder das Skeleton aufgenommen werden soll, fiel durch die oben genannten Gründe sehr einfach. Ein Video reicht aus, da die genauen Positionen nicht bekannt sein müssen.

Kapitel 4

Umsetzung

Das Kapitel Umsetzung beschreibt die Schritte der Applikationsentwicklung und die damit zusammenhängenden Überlegungen. Zum einen wird der Prototyp beschrieben und zum anderen die Weiterentwicklung, welche nach dem Prototyp entstand.

4.1 Prototyp

Der erste Prototyp entstand aus den Überlegungen der Winkelberechnungen, siehe Kapitel Konzeptionelle Überlegungen. Er enthielt die folgenden Funktionalitäten:

- Auswahl von verschiedenen Therapie Übungen
- Wählbare Einstellungen (Logdatei erstellen, Anzahl Übungswiederholungen)
- Neues Video für Übungsdurchführung aufnehmen
- Übung durchführen mit und ohne Video

Zusätzlich dazu wurde ein weiterer Prototyp erstellt, welcher mit Hilfe der Log-Dateien eine grafische Auswertung der Übungen durchführt. Im nachfolgendem Abschnitt sind die Mockups beschrieben, welche durch die Erkenntnisse von bestehenden Applikationen erstellt worden sind. Weiter wurde eine Domainanalyse durchgeführt, welche im weiteren Verlauf der Bachelorarbeit verwendet werden kann. Ebenfalls ist eine grobe Beschreibung der Implementation gegeben. Dieser bezieht sich auf die verschiedenen Bausteine, die verwendet wurden für die Entwicklung des Prototypen.

4.1.1 Mockups

Für den Prototyp wurden die nachfolgenden Mockups erstellt. Zuerst soll die Möglichkeit bestehen, eine Übung auszuwählen (siehe Abbildung 4.1).

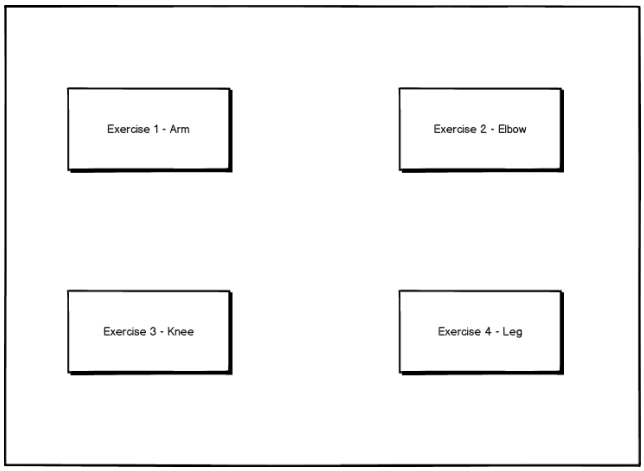


Abbildung 4.1: Mockup Prototyp: Hauptmenü

Danach können Konfigurationen vorgenommen werden. Diese beschränken sich auf die Möglichkeit die Daten der Übungsausführung in eine Log-Datei zu schreiben und ein neues Video aufzunehmen (siehe Abbildung 4.2).

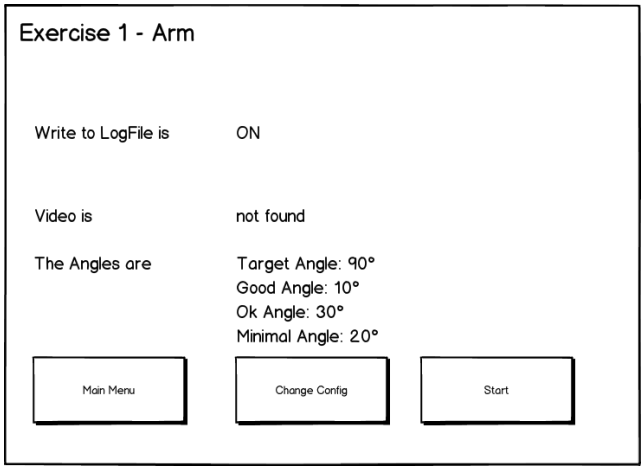


Abbildung 4.2: Mockup Prototyp: Konfiguration

Das Video wird in einer neuen Ansicht aufgenommen. Zuerst läuft ein Countdown ab, damit sich der Benutzer vorbereiten kann. Danach kann für eine gewisse Zeit aufgenommen werden (siehe Abbildung 4.3). Das Video wird für die gewählte Übung gespeichert und kann durch wiederholtes Aufzeichnen überschrieben werden.

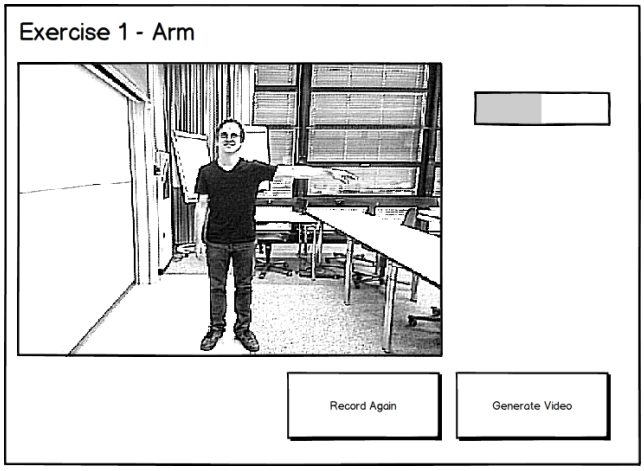


Abbildung 4.3: Mockup Prototyp: Video Aufnahme

Daraufhin hat der Benutzer die Wahl, die Übung mit oder ohne Video durchzuführen (siehe Abbildung 4.4 und 4.5).

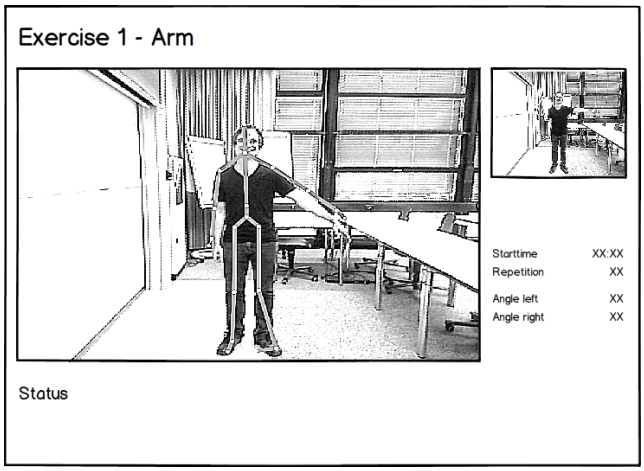


Abbildung 4.4: Mockup Prototyp: Übungsdurchführung mit Video

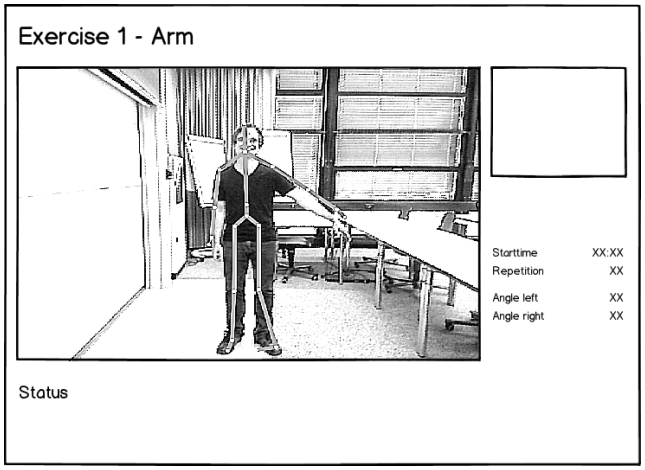


Abbildung 4.5: Mockup Prototyp: Übungsdurchführung ohne Video

Nach zehn Wiederholungen oder nach zehn Sekunden, in welcher die Kinect kein Skeleton erkennt, wird die Übung unterbrochen. Der Benutzer wird gefragt, ob er weiter machen oder zurück zum Hauptmenü will.

4.1.2 Domainmodel

Die Abbildung 4.6 zeigt das Domainmodel des Prototyps.

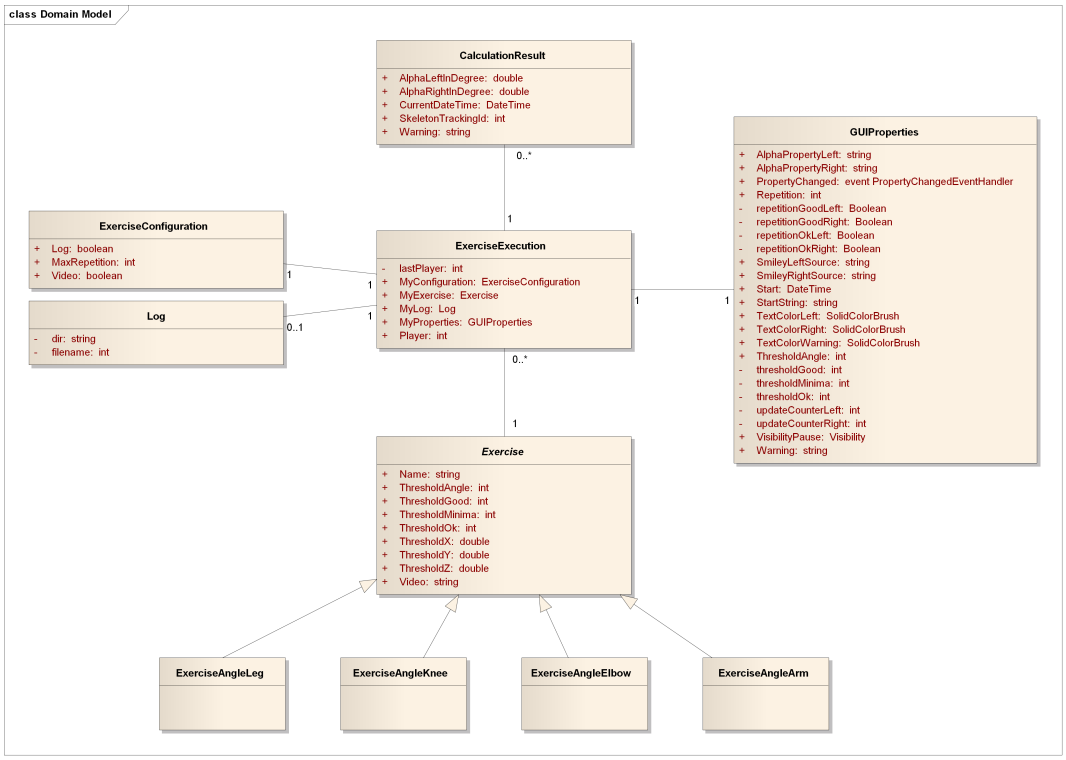


Abbildung 4.6: Domainmodel des Prototyps

Um den Prototypen möglichst generisch zu halten, wurde eine abstrakte Klasse «Exercise» definiert. Sie repräsentiert Übungen und hat folgenden Aufbau:

```
1  public abstract class Exercise
2  {
3      public int ThresholdAngle { get; set; }
4      public int ThresholdGood { get; set; }
5      public int ThresholdOk { get; set; }
6      public int ThresholdMinima { get; set; }
7      public double ThresholdX { get; set; }
8      public double ThresholdY { get; set; }
9      public double ThresholdZ { get; set; }
10     public string Name { get; set; }
11     public string Video { get; set; }
12     public abstract CalculationResult CalculateAngle(Skeleton
           skeleton);
13     public abstract string GetWarning(Skeleton skeleton);
14 }
```

Listing 4.1: Abstrakte Klasse «Exercise»

Wie im Listing 4.1 aufgezeigt, besteht die abstrakte Klasse aus Properties und zwei Methoden. Die Properties werden zur Auswertung sowie zur Beschreibung benötigt. Die zwei Methoden «CalculateAngle(..)» und «GetWarning(..)» dienen als Platzhalter für die eigentliche Implementation. Als Rückgabewert der Methode «CalculateAngle(..)» wird ein «CalculationResult»-Objekt verwendet. Dieses enthält einen Zeitstempel, einen Warnungstext, zwei Winkel sowie die SkeletonTrackingId (siehe auch Listing 4.2).

```
1  public class CalculationResult
2  {
3      public DateTime CurrentDateTime { get; set; }
4      public string Warning { get; set; }
5      public double AlphaRightInDegree { get; set; }
6      public double AlphaLeftInDegree { get; set; }
7      public int SkeletonTrackingId { get; set; }
8
9      public CalculationResult(DateTime currentDateTime, string
           warning, double alphaLeftInDegree, double
           alphaRightInDegree, int skeletonTrackingId)
10     {
11         // Zuweisung der Parameter an die Properties
12     }
13 }
```

Listing 4.2: Klasse «CalculationResult»

Eine konkrete Übung implementiert die von der abstrakten Klasse definierten Methoden. Für die Zuweisung der Variablen wird eine vordefinierte XML-Datei ausgelesen. Diese beinhaltet die Werte für die jeweilige Übung. Soll ein Wert abgeändert werden, kann lediglich die entsprechende XML-Datei angepasst werden. Die Applikation muss hierfür nicht neu kompiliert werden. Im Listing 4.3 wird ein Beispiel für die XML-Datei gezeigt, sowie ein verkürzter Auszug einer konkreten Implementierung im Listing 4.4.

```
1  <ExerciseMetadata>
2    <thresholdAngle>90</thresholdAngle>
3    <thresholdGood>10</thresholdGood>
4    <thresholdOk>30</thresholdOk>
5    <thresholdMinima>20</thresholdMinima>
6    <thresholdX>0</thresholdX>
7    <thresholdY>0.1</thresholdY>
8    <thresholdZ>0.1</thresholdZ>
9    <name>Arm</name>
10   <video>C:\temp\20130411_113539.avi</video>
11 </ExerciseMetadata>
```

Listing 4.3: Beispiel XML-Datei der Arm Übung

```
public class ExerciseAngleArm : Exercise
{
    public override CalculationResult CalculateAngle(Skeleton
        skeleton)
    {
        // konkrete Implementation
    }
    public override string GetWarning(Skeleton skeleton)
    {
        // konkrete Implementation
    }
}
```

Listing 4.4: Klasse «ExerciseAngleArm»

Um eine Übung durchzuführen, braucht es eine Übungskonfiguration (realisiert mit der «ExerciseConfiguration»-Klasse, siehe Listing 4.5) sowie eine konkrete Übung. Dies wird in der Klasse «ExerciseExecution» (siehe Listing 4.6) vereint. Gleichzeitig wird diese Klasse verwendet, um die Berechnung anzustossen, sowie die entsprechenden Ergebnisse anzeigen zu lassen.

```
public class ExerciseConfiguration
{
    public int MaxRepetition { get; set; }
    public Boolean Log { get; set; }
    public Boolean Video { get; set; }
    public ExerciseConfiguration(int maxRepetition, Boolean log
        , Boolean video)
    {
        // Zuweisung der Parameter an die Properties
    }
}
```

Listing 4.5: Klasse «ExerciseConfiguration»

```
public class ExerciseExecution
{
    private int player = -1;
    private DateTime lastPlayer;
    public int Player
    {
        get { return player; }
        set
        {
            if (value != -1 && player != value)
            {
                SetStart(DateTime.Now);
            }
            lastPlayer = DateTime.Now;
            player = value;
        }
    }
    public GUIProperties MyProperties { get; set; }
```

```
19      public Log MyLog { get; set; }
20      public ExerciseConfiguration MyConfiguration { get; set; }
21      public Exercise MyExercise { get; set; }
22
23      public ExerciseExecution(ExerciseConfiguration
            myConfiguration, Exercise exercise)
24      {
25          MyConfiguration = myConfiguration;
26          MyExercise = exercise;
27          MyProperties = new GUIProperties(MyExercise.
                thresholdAngle, MyExercise.thresholdGood,
                MyExercise.thresholdOk, MyExercise.thresholdMinima)
                ;
28      }
29      public void SetStart(DateTime dateTime)
30      {
31          MyProperties.ResetPlayer();
32          MyProperties.Start = dateTime;
33          MyLog = new Log(dateTime);
34      }
35      private Boolean IsPlayer(int trackingId)
36      {
37          if (Player == -1 || Player == trackingId)
38          {
39              Player = trackingId;
40              return true;
41          }TimeSpan timeDiff = DateTime.Now - lastPlayer;
42          if (timeDiff.TotalSeconds > 5)
43          {
44              Player = trackingId;
45              return true;
46          }return false;
47      }
48      public async void Calculate(Skeleton skeleton)
49      {
50          Task<CalculationResult> t1 = Task.Run(() => MyExercise.
                CalculateAngle(skeleton));
51          CalculationResult calcResult = await t1;
52          if (IsPlayer(calcResult.SkeletonTrackingId))
53          {
54              MyProperties.UpdatePlayer(calcResult.
                    AlphaLeftInDegree, calcResult.
                    AlphaRightInDegree, calcResult.Warning);
55              if (MyConfiguration.Log && MyLog != null)
56              {
57                  MyLog.WriteLog(calcResult.AlphaLeftInDegree,
                        calcResult.AlphaRightInDegree, calcResult.
                        CurrentDateTime, MyProperties.
                        ThresholdAngle);
58              }
59          }
60      }
61  }
```

Listing 4.6: Klasse «ExerciseExecution»

4.1.3 Implementation

Die Implementation des Prototypen teilt sich in zwei Bereiche. Zum einen wurde eine Applikation zur Übungsentwicklung erstellt. Zum anderen eine, welche die Übungen mit Hilfe der Log-Dateien grafisch auswertet.

Übungsdurchführung

Zuerst wurde überlegt, wie die Applikation am Besten über die Kinect gesteuert werden kann. Für die Bedienung sollte keine Zweitperson oder Maus benötigt werden. Dafür bietet das Kinect SDK v1.7 eine Lösung. Die «Kinect Interaction, push» Funktion ermöglicht das Steuern einer Applikation mit der Hand (siehe Kapitel Grundlagen, Abschnitt Kinect-Software Developement Kit). Der Benutzer kann einen Button mit der Hand klicken. Diese Erweiterung wurde vom Beispiel «Control Basics» übernommen, welches Microsoft mit dem SDK bereitstellt. Die Funktion läuft sehr flüssig und kann für die Steuerung der Applikation sehr gut benutzt werden (siehe Abbildung 4.7).

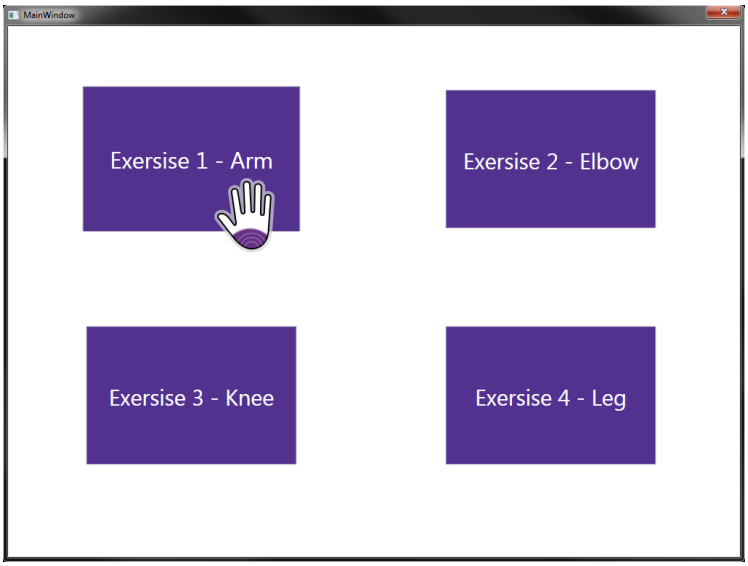


Abbildung 4.7: Screenshot Prototyp: Hauptmenü

Als Übungen sind die Winkelberechnungen aus dem Kapitel Konzeptionelle Überlegungen angepasst und verwendet worden. Wie bereits im Domainmodel erläutert, können diese einfach ausgetauscht werden. Die Abbildung 4.8 zeigt einen Screenshot einer Übungsdurchführung.

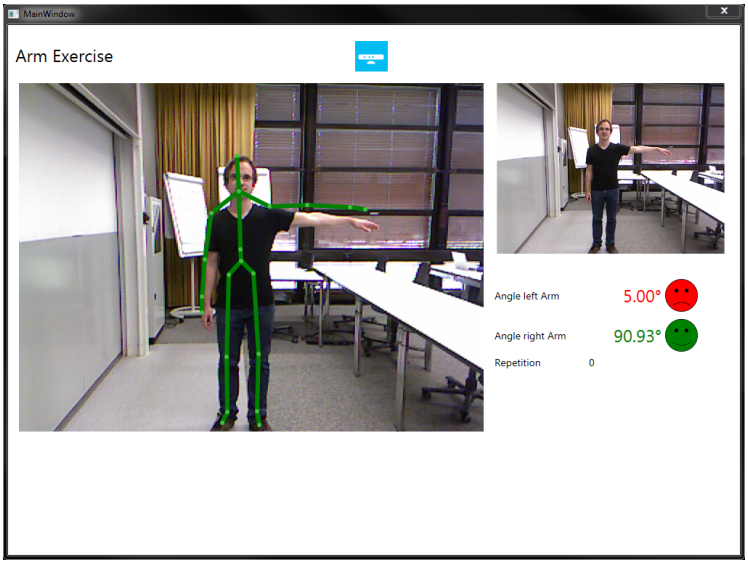


Abbildung 4.8: Screenshot Prototyp: Übungsdurchführung

Vor der Übungsdurchführung kann ausgewählt werden, ob der Benutzer mit oder ohne Aufnahme weiterfahren will. Die Aufnahme eines Übungsvideo wird nach einem Countdown von fünf Sekunden ausgelöst und kann beliebig wiederholt werden (siehe Abbildung 4.9). Dies ist ein Bestandteil des Prototyps, welcher je nach Zielpublikum genau wie die Übungen einfach angepasst werden kann. Die Aufnahme wird in einer Endlosschlaufe als Video abgespielt. Die Videogenerierung dauert einige Zeit, da viele Einzelbilder zu einem Video zusammengefügt werden müssen.

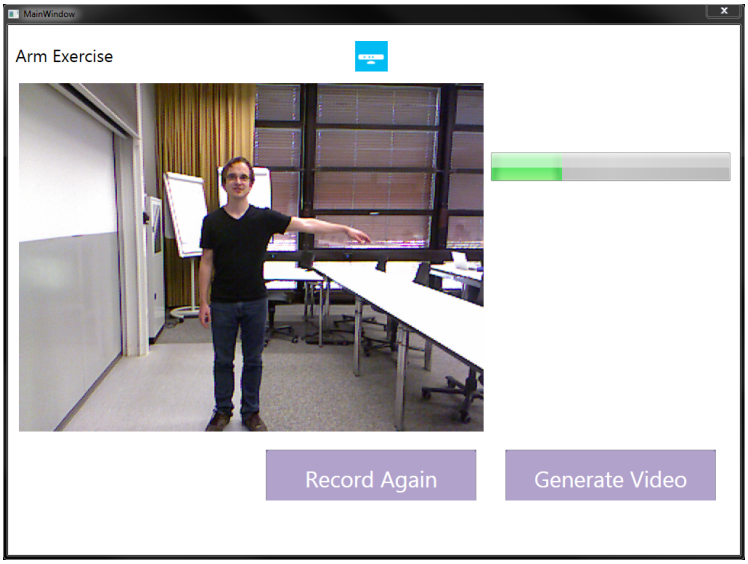


Abbildung 4.9: Screenshot Prototyp: Video Aufnahme

Der Prototyp für die Übungsdurchführung war lauffähig und hat die geforderten Funktionalitäten angeboten. Schnell wurde jedoch erkannt, dass Einschränkungen bestehen. So wird beispielsweise die Winkelgrösse beim Erstellen einer Übung festgelegt und kann nachträglich nicht in der Applikation geändert werden. Je nach Bewegungseinschränkung eines Benutzers, ist es schwierig, den gewünschte Winkel zu erreichen und somit die Übung richtig durchzuführen. Der Prototyp wurde mit einer weiteren Funktion ergänzt. Diese bietet die Möglichkeit, die Winkel selber einzustellen. Dies wurde über einen Zahlenblock ermöglicht, sodass die Werte gemäss den Vorgaben des Therapeut eingegeben werden können (siehe Abbildung 4.10).

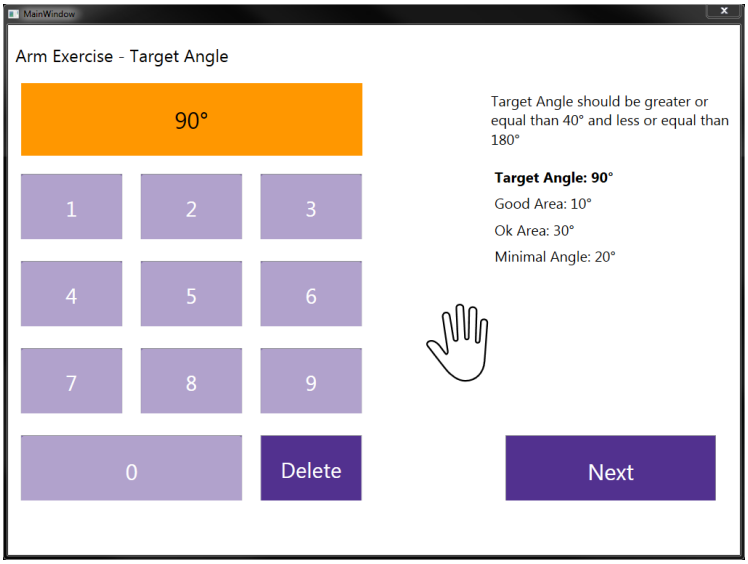


Abbildung 4.10: Screenshot Prototyp: Zahlenblock

Übungsauswertung

Neben dem Prototyp für die Übungsdurchführung soll auch die Möglichkeit bestehen, die Übungen auszuwerten. Hierfür eignet sich eine einfache grafische Darstellung der Messergebnisse am Besten.

In einem ersten Versuch wurden die Ergebnisse (berechnete Winkel) der Winkelberechnung in eine Datei geschrieben. Diese könnte im Anschluss dem Arzt übermittelt und analysiert werden. Das Ziel des Prototyp war, die Daten mit einer zweiten Applikation zu lesen und grafisch darzustellen. Die grafische Darstellung wurde mit einem Liniendiagramm realisiert. Dieses beinhaltet die berechneten Winkel des linken sowie des rechten Arms, sowie den aktuellen Winkel, welcher erreicht werden sollte.

Da eine medizinische Übung meistens über eine längere Zeitdauer durchgeführt wird, kann dies zu einer längeren Zeitachse führen. Wird nun eine feste Breite für das Diagramm gewählt, so kann dieses unübersichtlich werden, da zu viele Daten auf eine zu kleinen Breite dargestellt werden müssen. Die Breite respektive der Zeitausschnitt muss deshalb variabel wählbar sein. Wird eine lange Aufzeichnung angeschaut, so vergrössert sich die Grafik in der Breite.

Die Abbildung 4.11 zeigt die Darstellung des Prototypen.

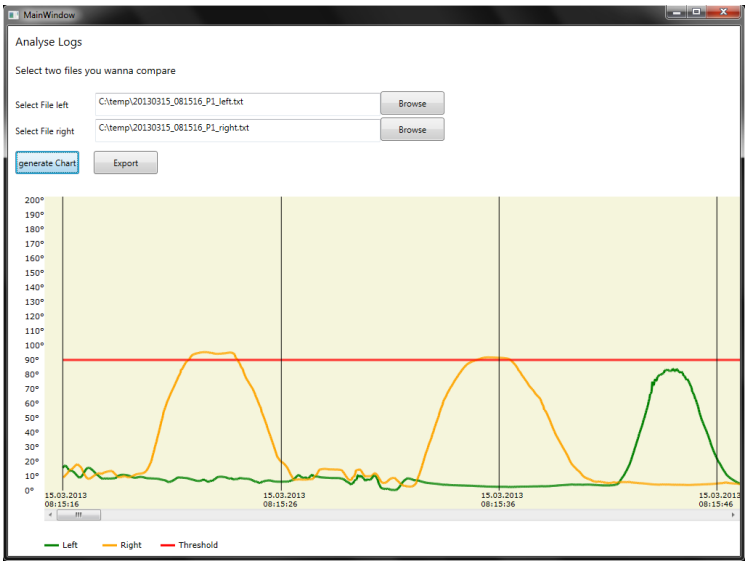


Abbildung 4.11: Screenshot: grafische Darstellung einer Übungsdurchführung

Die Realisierung des Liniendiagramms wurde mit Punkten und Verbindungslinien erreicht. So kommt es, dass ziemlich viele Elemente auf die Oberfläche gezeichnet werden. Da diese alle über den «UI-Thread» laufen, kommt es dazu, dass die Anwendung «einfriert» bis sämtliche Elemente gezeichnet sind. Die Applikation ist dabei mit dem Zeichnen der Linien und Punkte beschäftigt, sodass ein Ereignis, wie beispielsweise «verschieben des Fensters», nicht sofort abgearbeitet wird. Dies ist besonders bei längeren Aufzeichnungen bemerkbar.

Dieses Einfrieren kann mit einem «Workaround» umgangen werden (Tipp von Prof.

Hansjörg Huser). Da bis zu mehrere 10'000 Linien oder Punkte gezeichnet werden müssen, kann gesagt werden, dass alle 1'000 Zeichnungsoperationen eine Methode «DoEvents()» aufgerufen werden soll. Diese Methode unterbricht den Zeichnungsprozess und bearbeitet andere Ereignisse wie «verschieben des Fensters». Die grafische Benutzeroberfläche, das GUI, erscheint dadurch flüssiger.

Der Code für die Methode «DoEvents()» ist im Listing 4.7 ersichtlich.

```
1  [SecurityPermissionAttribute(SecurityAction.Demand, Flags =
       SecurityPermissionFlag.UnmanagedCode)]
2  public void DoEvents()
3  {
4      DispatcherFrame frame = new DispatcherFrame();
5      Dispatcher.CurrentDispatcher.BeginInvoke(DispatcherPriority
           .Background,
6          new DispatcherOperationCallback(ExitFrame), frame);
7      Dispatcher.PushFrame(frame);
8  }
9
10 public object ExitFrame(object f)
11 {
12     ((DispatcherFrame)f).Continue = false;
13     return null;
14 }
```

Listing 4.7: Workaround Methode «DoEvents()» [13]

Wird der Code mit diesem Workaround ergänzt, so kann das Fenster verschoben werden und die Applikation reagiert genügend schnell auf solche Ereignisse. Der Workaround hat aber gleichzeitig auch zur Folge, dass das Zeichnen der Grafik vermehrt unterbrochen wird und ein Wechsel auf die DoEvents()-Methode stattfindet. Dieser Wechsel führt natürlich zu einem Performanceverlust, so dass ein geeigneter Wert, wie oft aktualisiert werden soll, noch gefunden werden muss.

4.2 Probleme und Erkenntnisse durch den Prototypen

Der Prototyp besteht aus zwei lauffähigen Applikationen. Die erste Applikation erlaubt es, Übungen durchzuführen und ein Anleitungsvideo der Übung zu erstellen. Die zweite wertet die Daten einer Übungsdurchführung grafisch aus.

Wie bereits im Kapitel Konzeptionelle Überlegungen, Abschnitt Übungsdurchführung, erwähnt, sind die Möglichkeiten sehr beschränkt. Es können zwar verschiedene Gelenke betrachtet und deren Winkel berechnet werden, allerdings müssen diese jeweils selber programmiert werden. Dies schränkt das Einsatzgebiet der Applikation weiter ein.

Die Anforderung, dass selber Übungen erstellt werden können, ist damit nicht erfüllt. Zusätzlich wird es schwierig, sobald mehrere Winkel beachtet werden sollen. Weitaus schöner wäre eine Variante, bei welcher jeder Anwender eine Übung erstellen kann. Im Abschnitt Generische Position der Gelenke wurden die konzeptionellen Überlegungen beschrieben, wie eine solche Lösung angestrebt werden kann.

Aus Sicht der Softwareentwicklung bietet der Prototyp ebenfalls Optimierungspotential. Der Code selbst befindet sich im «Code-Behind». Es besteht somit keine klare Trennung zwischen Code und Anzeige.

Nach einigen Test mit Mitstudenten hat sich gezeigt, dass der Programmverlauf nicht offensichtlich ist. Aussenstehende Testpersonen konnten die Applikation bedienen, jedoch erst nach zusätzlichen Hinweisen, was genau zu tun ist. Es müsste die Möglichkeit bestehen, spezifische Angabe zum nächsten Schritten anzuzeigen. Gemäss der Aufgabenstellung der Bachelorarbeit soll die Applikation benutzerfreundlich und einfach bedienbar sein. So gesehen war dies ein wichtiger Hinweis der Mitstudenten.

Im Abschnitt Weiterentwicklung wird der Prototyp gemäss der gewonnenen Erkenntnissen angepasst und verbessert.

4.3 Weiterentwicklung

Wie bereits im vorhergehenden Abschnitt und im Kapitel Konzeptionelle Überlegungen beschrieben, sind die Winkelberechnungen sehr beschränkt einsetzbar. Es wurde entschieden die generischen Übungen einzusetzen. Dadurch musste die ganze Applikation angepasst und umstrukturiert werden. Die Ansichten mit den Winkeleinstellungen werden nicht mehr benötigt. Ebenfalls wird das MVVM-Konzept (Abkürzung für Model View ViewModel) umgesetzt, welches im Kapitel Softwarearchitektur beschrieben wird. Gleichzeitig wird die Applikation in eine Vollbild-Applikation umgeschrieben, sodass diese auch auf einem grossen Bildschirm (wie beispielsweise einem Fernseher) gestartet werden kann.

Auch das Hauptmenü musste verändert werden. Aufgrund der generischen Übungen, soll die Möglichkeit bestehen, beliebig viele Übungen zu erstellen. Für jede erstellte Übung muss ein Button im Hauptmenü vorhanden sein. Dies konnte mit einer «ObservableList» gelöst werden. Diese Liste meldet dem User Interface, wenn ein weiterer Button der Liste hinzugefügt wurde. Das Hauptmenü kann dadurch zur Laufzeit erweitert werden, wenn eine neue Übung erstellt wird.

Ebenso wurde im Laufe der Entwicklung die Konfigurationsseite als eher störend betrachtet. Zukünftig wird zu jeder Übung eine Logdatei geschrieben. Eine ähnliche Erkenntnis konnte bei der Anzeige des Videos gefunden werden. Damit der Anwender die Übung verstehen kann, muss er wissen, wie die Übung durchgeführt werden sollte. So kommt es, dass der Anwender vor jeder Übung das Video anschauen muss, bevor diese gestartet werden kann. Auch während der Übung wird das Video ständig wiederholt. Zusätzlich muss jedes mal wenn eine neue Übung erstellt wird, zwingend ein Video aufgenommen werden. Dadurch fällt die Möglichkeit weg, zu einem späteren Zeitpunkt ein eigenes Video aufzunehmen. Die einzelnen Schritten wurden mit Beschreibungen ergänzt, welche Hinweise zur Durchführung geben.

Das Domainmodel wird im Kapitel Softwarearchitektur genau erläutert.

4.3.1 Mockups

Im nachfolgendem Abschnitt sind die Mockups der entwickelten Applikation beschrieben. Diese zeigen den Ablauf die Abläufe der Applikation auf. Überlegungen und Probleme werden im Abschnitt Implementation erläutert und beschrieben.

Wie bereits beim Prototyp soll auch bei der Weiterentwicklung die Möglichkeit bestehen, eine Übung auszuwählen. Zusätzlich muss jedoch eine neue Übung erstellt werden können. Die Abbildung 4.12 zeigt das entsprechende Hauptmenü

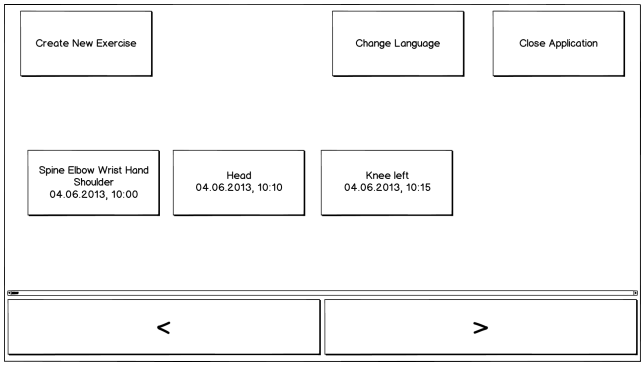


Abbildung 4.12: Mockup: Hauptmenü

Die Applikation hat somit zwei mögliche Abläufe: Übung auswählen und durchführen oder eine neue erstellen. Wird eine neue Übung gemäss den Überlegungen von generischen Übungen erstellt, so müssen einzelne Teilschritte aufgenommen werden. Hierfür wird eine neue Ansicht benötigt (siehe Abbildung 4.13).

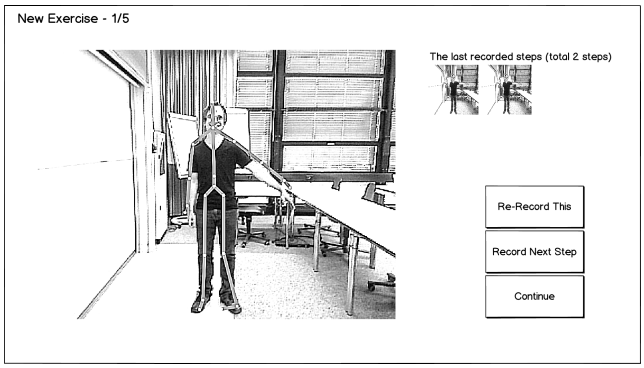


Abbildung 4.13: Mockup: neue Übung erstellen

Nach dem die einzelnen Schritte aufgenommen worden sind, müssen weiter die Achsen und Gelenke gewählt werden. Die Abbildungen 4.14, 4.15 und 4.16 zeigen die Ansichten.

Anders als im Kapitel Grundlagen beschrieben, werden nur 19 Gelenkpunkte zur Auswahl gegeben. Dies ist darauf zurückzuführen, dass der Punkt des Schulterzentrums ein berechneter Wert aus Schulter links und rechts ist. Es handelt sich dabei nicht um einen getrackten Punkt, sodass dieser bei der Auswahl weggelassen wurde (siehe Abbildung 4.15).

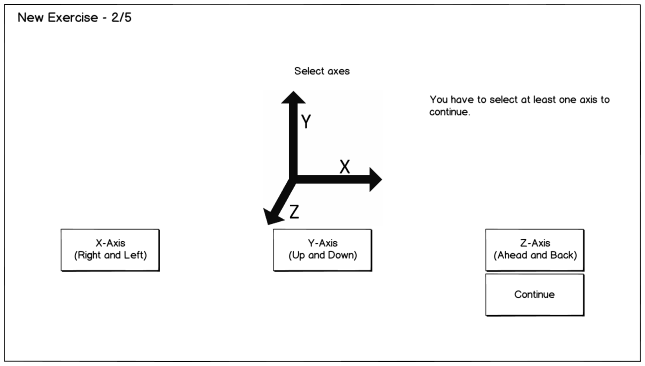


Abbildung 4.14: Mockup: Involvierte Achsen der neuen Übung auswählen

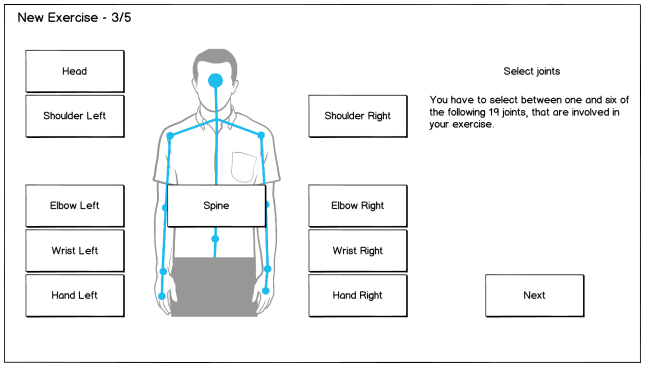


Abbildung 4.15: Mockup: Involvierte Gelenke der neuen Übung auswählen (Oberkörper)

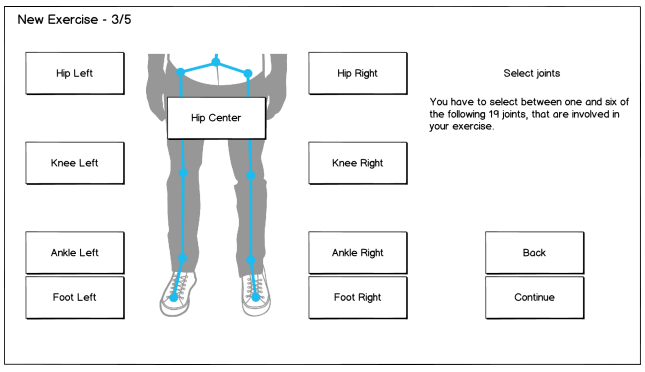


Abbildung 4.16: Mockup: Involvierte Gelenke der neuen Übung auswählen (Unterkörper)

Zum Abschluss soll ein Video aufgenommen und die Übung gespeichert werden (siehe Abbildung 4.17 und 4.18).

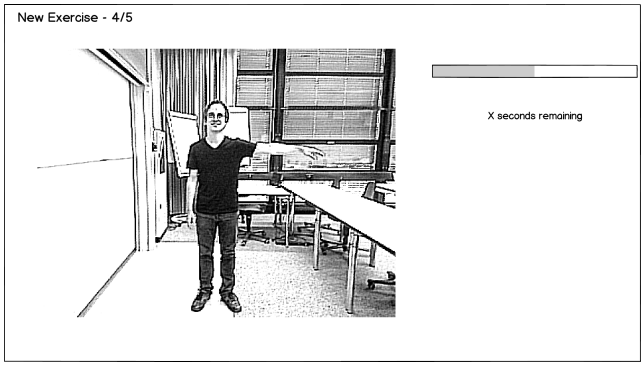


Abbildung 4.17: Mockup: Video Aufnahme

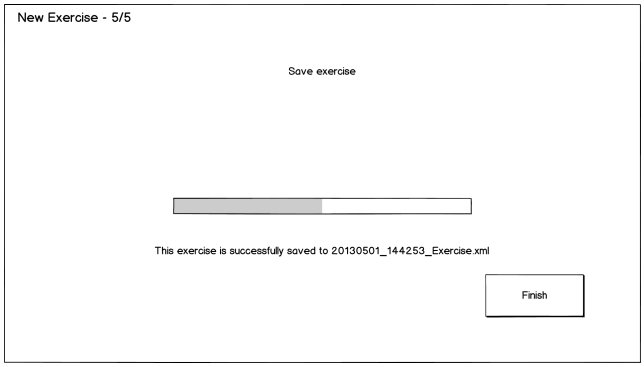


Abbildung 4.18: Mockup: neue Übung speichern

Nach diesen Schritten ist die Übung erstellt. Der Ablauf der Übungsdurchführung hat sich im Vergleich zum Prototyp verändert. Wie erwähnt muss der Benutzer zwingend das Video zur Übung anschauen (siehe 4.19).

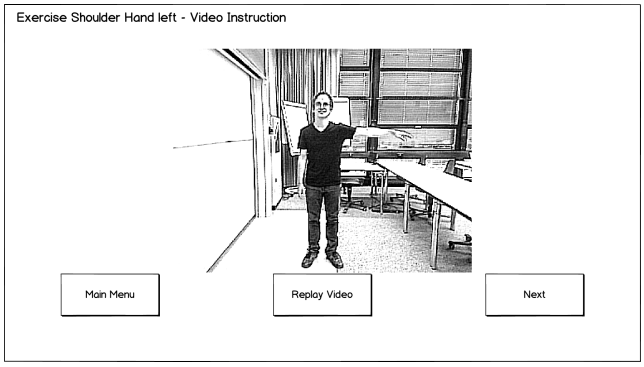


Abbildung 4.19: Mockup: Video Anleitung der Übung

Daraufhin kann der Benutzer die Übung durchführen. Im Vergleich zum Prototypen

werden nicht mehr die Winkel angegeben, sondern die einzelnen Teilschritte der Übung (siehe Abbildung 4.20).

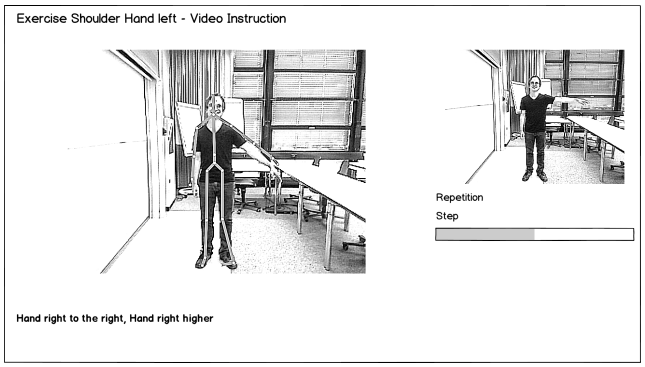


Abbildung 4.20: Mockup: Übungsdurchführung

Wie bereits im Prototyp soll die Möglichkeit bestehen, die Übung nach einer gewissen Zeit zu unterbrechen (siehe Abbildung 4.21).

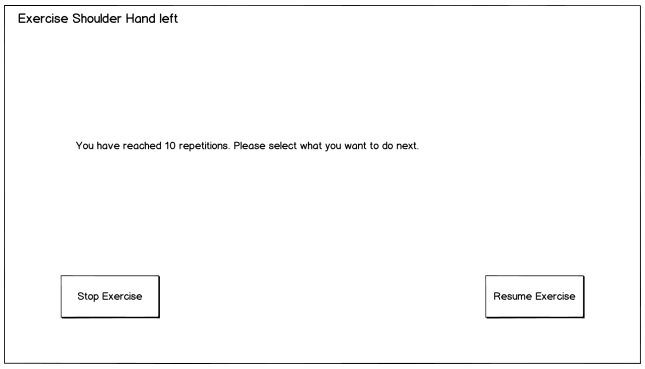


Abbildung 4.21: Mockup: Unterbrechung der Übungsdurchführung (Anzahl Wiederholungen erreicht)

4.3.2 Implementation

Anpassung der Ansichten

Da die Möglichkeit nun besteht eine Übung selber zu erstellen, kann es dementsprechend sehr viele Buttons für die erstellten Übungen geben. Deshalb musste überlegt werden, welches Konzept man anwenden soll. Die erste Überlegung war, mehrere Seiten zu erstellen. Dies wäre allerdings mühsam, wenn eine Übung gesucht werden muss. Es müsste auf mehreren Seiten nachgesehen werden. Als Alternative wurde der «KinectScrollViewer» in Kombination mit den «KinectTileButton» entdeckt. Dies wurde im Beispiel «Control Basics» von Microsoft verwendet. Die Bedienung erscheint angenehm, sodass entschieden wurde, diese Kombination zu verwenden.

Die neuen Übungen werden in einer «ObservableList» gespeichert. Das Hauptmenü kann dadurch zur Laufzeit erweitert werden, wenn eine neue Übung erstellt wird. Das Hauptmenü soll weiter die Möglichkeit bieten, die Sprache zu ändern. Dafür wurde ein weiterer Button erstellt. Der Benutzer kann zwischen Deutsch und Englisch wählen. Aus diesen Überlegungen entstand das Hauptmenü, wie es in Abbildung 4.22 dargestellt wird.

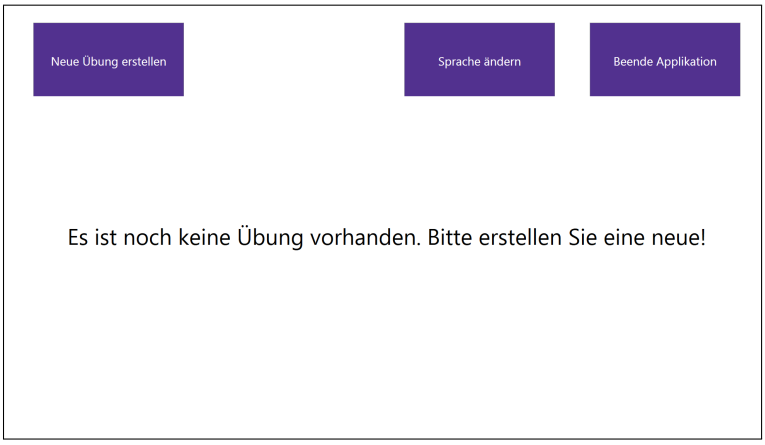


Abbildung 4.22: Screenshot: Hauptmenü (keine Übungen vorhanden)

Will der Benutzer eine neue Übung erstellen, so muss zuerst der Körper kalibriert und damit ausgemessen werden. Dies erfolgt nach dem Ablauf eines Countdowns von fünf Sekunden. Während diesen fünf Sekunden kann sich die Person richtig positionieren. Anschliessend werden auf der gleichen Ansicht die Schritte der Übung aufgezeichnet. Der Benutzer sollte sich möglichst nicht bewegen, sofern es nicht Teil der Übung ist. Eine Übung besteht aus mindestens zwei Schritten. Die aufgenommenen Schritte werden oben rechts angezeigt. Bei Problemen beziehungsweise einem fehlerhaften Schritt kann der zuletzt Aufgenommene durch eine neue Aufnahme ersetzt werden. Vor jeder Aufnahme erhält der Benutzer fünf Sekunden Zeit für die Vorbereitung. Die Abbildung 4.23 zeigt die Ansicht, welche die Schritte aufnimmt.

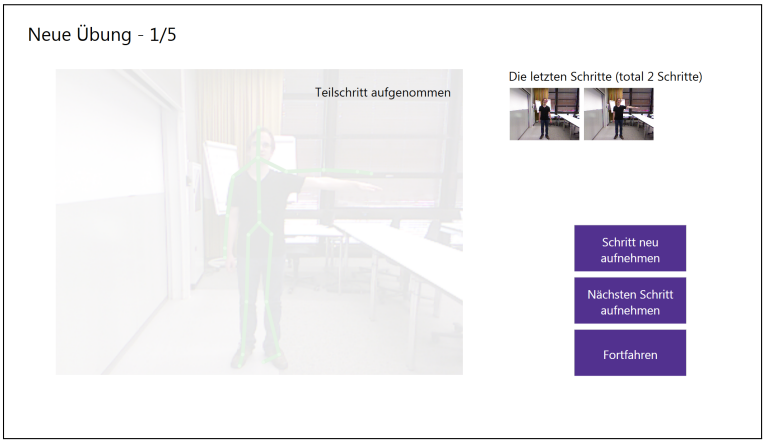


Abbildung 4.23: Screenshot: neue Übung erstellen

Falls eine andere Person im Bild erscheint oder die Kinect etwas anderes als Person erkennt, kommt es zu falschen Werten. Um dies bei der Kalibrierung und der Aufnahme der Schritte zu vermeiden, wurde beschlossen, dass die Kinect immer nur eine Person tracken soll. Dies wurde in allen anderen Ansichten, welche das Skeleton anzeigen, ebenfalls eingesetzt.

Die nächste Ansicht stellt die Auswahl der Achsen zu Verfügung. Wie bereits im Kapitel Konzeptionelle Überlegungen, Abschnitt Generische Position der Gelenke, erläutert wurde, kann ausgewählt werden, in welche Richtung die Bewegungen überprüft werden soll. Es muss mindestens eine Achse gewählt werden, um mit der Erstellung fortzufahren. Um klarer zu machen, was die X-, Y- und Z- Achsen bedeuten, wurden diese mit einem Bild und der Beschriftung der Buttons weiter erläutert (siehe Abbildung 4.24).

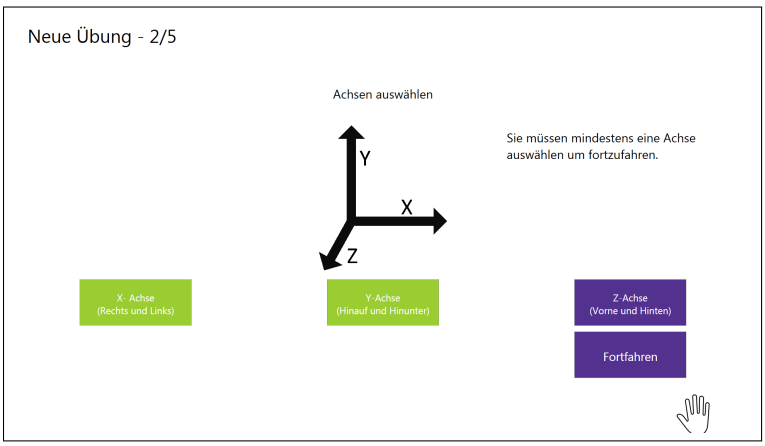


Abbildung 4.24: Screenshot: Involvierte Achsen der neuen Übung auswählen

Auf einer weiteren Ansicht können die Gelenke gewählt werden. Die Ansicht besteht aus zwei Teilen, dem Oberkörper und dem Unterkörper. Die beiden Abbildungen 4.25 und 4.26 zeigen diese Aufteilung. Auch hier muss mindestens eines von den 19 verfügbaren

gewählt werden. Diese Auswahl wurde jedoch auf maximal sechs Gelenke begrenzt, um die Komplexität der Übungsdurchführung zu begrenzen.

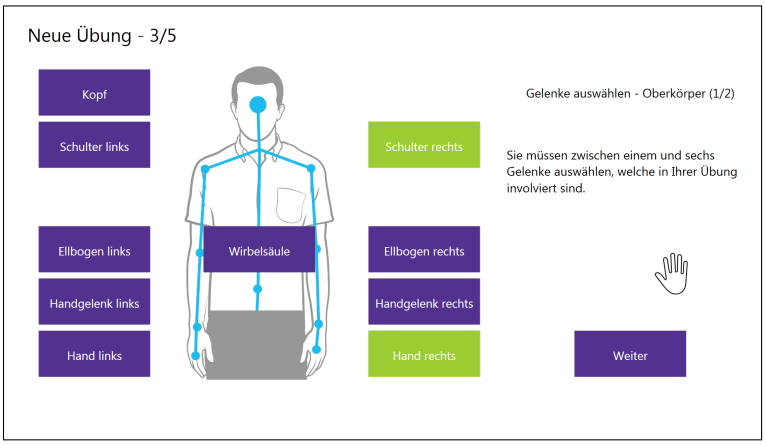


Abbildung 4.25: Screenshot: Involvierte Gelenke der neuen Übung auswählen (Oberkörper)

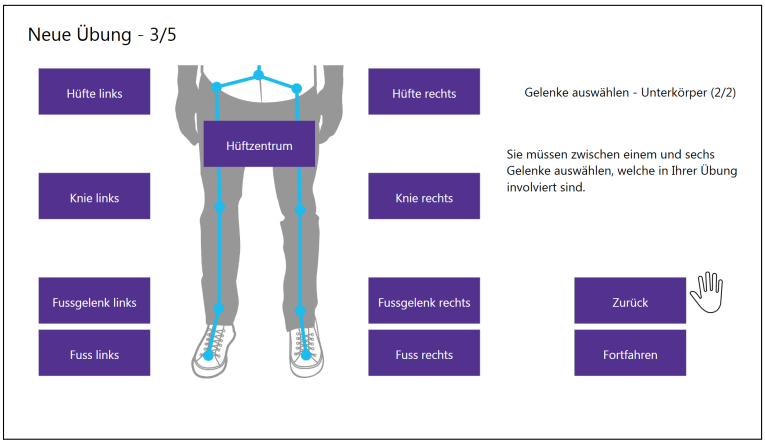


Abbildung 4.26: Screenshot: Involvierte Gelenke der neuen Übung auswählen (Unterkörper)

Um das Verständnis der Übung zu verbessern, muss zwingend ein Video aufgenommen werden. Wie bereits weiter oben erwähnt, muss klar sein, wie die Übung ausgeführt werden muss. Das Video dauert genau acht Sekunden. Während der Aufnahme wird die verbleibende Zeit wird mit einem Fortschrittsbalken und einem Countdown aufgezeigt (siehe Abbildung 4.27). Das Generieren des Videos dauert einen Moment. Während der Aufnahme werden die Frames, welche von der Kinect aufgenommen werden, in eine Liste zwischengespeichert. Die einzelnen Bilder aus dieser Liste werden daraufhin zu einem Video zusammengeschnitten und das dauert einige Zeit. Im Vergleich zum Prototyp wurde die Anzeige mit einem Fortschrittsbalken ergänzt (siehe Abbildung 4.28). Der Benutzer kann sich somit sicher sein, dass die Applikation arbeitet und nicht abgestürzt ist.

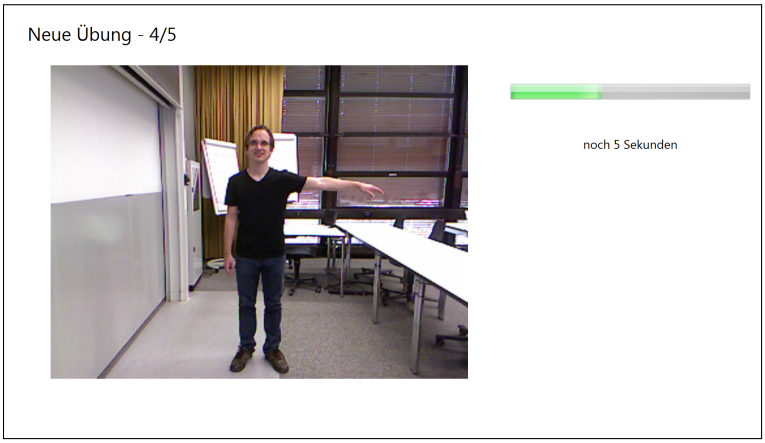


Abbildung 4.27: Screenshot: Video Aufnahme

Abbildung 4.28: Screenshot: Video Generierung

Zum Schluss wird die neu erstellte Übung gespeichert (siehe Abbildung 4.29). Dabei werden die gesammelten Daten zusammengetragen und in einer XML-Datei abgespeichert. Ein verkürzter Ausschnitt wird im Listing 4.8 gezeigt. Die XML-Datei besteht sowohl aus allgemeinen Angaben wie Name, Datum und Name des Videos sowie aus einem oder mehreren Übungsschritten (im XML als «GenericStep» bezeichnet). Im aufgezeigten Übungsschritt wurde zwei «GenericPosition's» gespeichert. Diese enthalten die relativen X- und Y-Positionen des rechten Ellbogen und der rechten Hand. Die Z-Position sollte nicht aufgezeichnet werden und ist deshalb in der XML-Datei als «nicht definiert» (abgekürzt n. def.) gekennzeichnet.

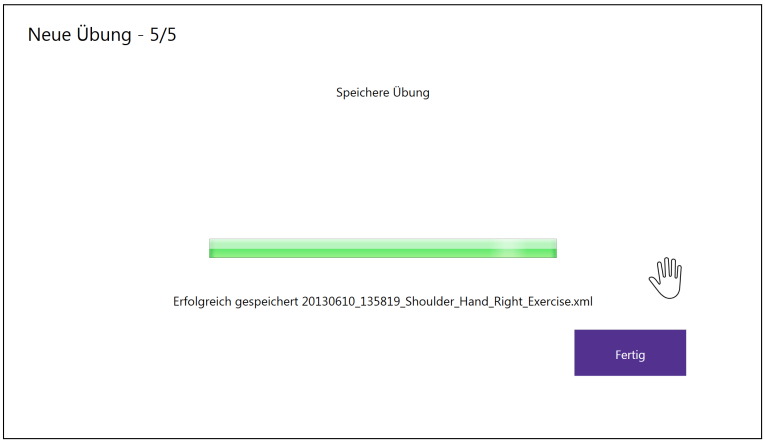


Abbildung 4.29: Screenshot: neue Übung speichern

```
<Generic>
  <Name>Shoulder_Hand_Right</Name>
  <Date>635064694997909620</Date>
  <Video>20130610_135819.avi</Video>
  <GenericStep>
    <Positions>
      <GenericPosition>
        <Joint>ShoulderRight</Joint>
        <PercentageX>57.7692007825601</PercentageX>
        <PercentageY>71.3471878125992</PercentageY>
        <PercentageZ>n. def.</PercentageZ>
      </GenericPosition>
      <GenericPosition>
        <Joint>HandRight</Joint>
        <PercentageX>63.0672062505246</PercentageX>
        <PercentageY>34.946908976996</PercentageY>
        <PercentageZ>n. def.</PercentageZ>
      </GenericPosition>
    </Positions>
  </GenericStep>
  [...]
</Generic>
```

Listing 4.8: Beispiel XML-Datei einer generischen Übung

Eine Übung besteht aus einem Ordner mit einem Zeitstempel, welcher die XML-Datei sowie die Video-Datei beinhaltet. Zusätzlich wird bei der ersten Übungsdurchführung der Ordner «Logs» erstellt.

Im Hauptmenü ist diese neu erstellte Übung sofort verfügbar (siehe Abbildung 4.30). Damit ein Benutzer die Übung wiedererkennen kann, sollen die Buttons der Übungen sinnvoll beschrieben werden. Auf dem Button werden die gewählten Gelenke sowie das Datum und die Uhrzeit der Erstellung angezeigt. Beschränkt sich die Übung auf eine Körperseite, so wird zusätzlich «links» beziehungsweise «rechts» angegeben.

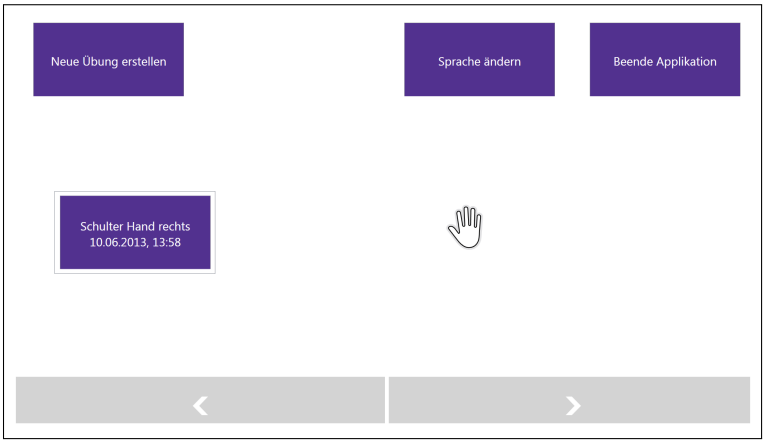


Abbildung 4.30: Screenshot: Hauptmenü (mit der neuen Übung)

Der Benutzer kann die Übung auswählen, wobei er die Möglichkeit hat, die Übung durchzuführen oder die Auswertungen vergangener Übungsdurchführungen ansehen (siehe Abbildung 4.31).

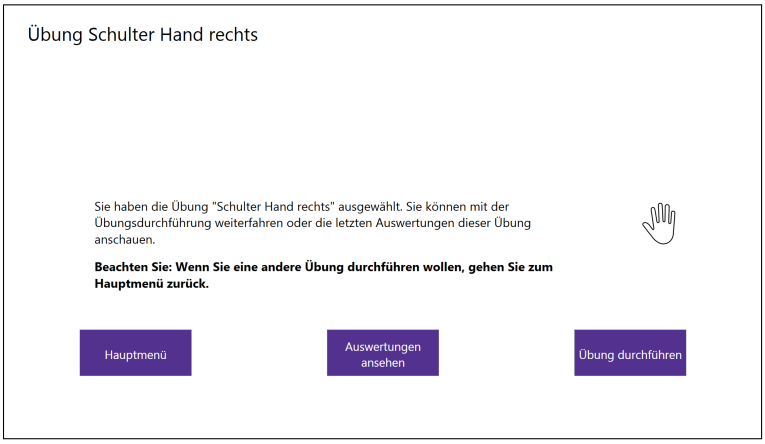


Abbildung 4.31: Screenshot: Übung ausgewählt, Möglichkeit Auswertung anzusehen oder Übung durchführen

Führt der Benutzer die Übung durch, so wird das entsprechende Video geladen und abgespielt (siehe Abbildung 4.32).

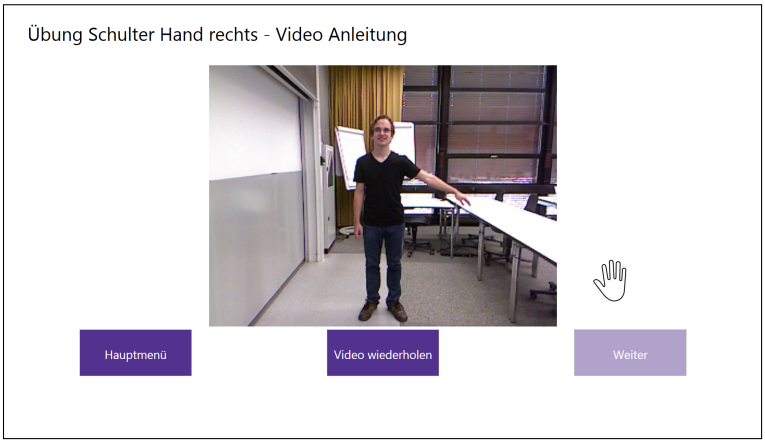


Abbildung 4.32: Screenshot: Video Anleitung der Übung

Sobald das Video mindestens einmal abgespielt wurde, kann der Benutzer mit der Übungsdurchführung fortfahren. Auch hier ist eine Kalibrierung des Körpers notwendig. So können die Positionen der Gelenke neu berechnet werden. Auf der Übungsansicht wird rechts oben das Video angezeigt. Der Benutzer weiss damit immer, wie die Übung durchgeführt werden soll. Darunter werden die Anzahl der Wiederholungen sowie die der Schritte. Letztere zeigt auf, wie viele Schritte der Benutzer richtig erreicht hat. Die Abbildung 4.33 zeigt die Ansicht der Übungsdurchführung.

Abbildung 4.33: Screenshot: Übungsdurchführung

Ebenfalls werden auf dieser Ansicht entsprechend «Anweisungen» und Korrekturen für die Erfüllung des Teilschrittes angegeben. Diese werden aufgrund des Vergleich der Positionswerte generiert. Das System zieht dabei einen Toleranzwert zwischen einem und drei Prozent ab. Ist in einer Übung lediglich ein Gelenk involviert, so beträgt der Toleranzwert eins, um die Übung möglichst präzis durchzuführen. Sind drei oder mehr Gelenke involviert, beträgt der Wert drei Prozent. In diesem Fall wird eine gewisse Toleranz erlaubt, da mehrere Gelenke gleichzeitig koordiniert werden müssen.

Ist ein Teilschritt korrekt, so muss der nächste Teilschritt angestrebt werden. Auch hier werden Meldungen zur Korrektur angezeigt. Sind alle Teilschritte erfüllt, wird eine Wiederholung dazu gezählt und die Teilschritte beginnen nochmals.

Wird der Körper mehr als zehn Sekunden nicht mehr erkannt, wird die Übungsdurchführung beendet und die Übung muss neu gestartet werden (siehe Abbildung 4.34). Hat der Benutzer die Anzahl Wiederholungen erreicht, hat er die Wahl, sie nochmals mit der Übung weiterzufahren oder die Auswertung der durchgeführten Übung anzusehen und die Übung zu beenden (siehe Abbildung 4.35).

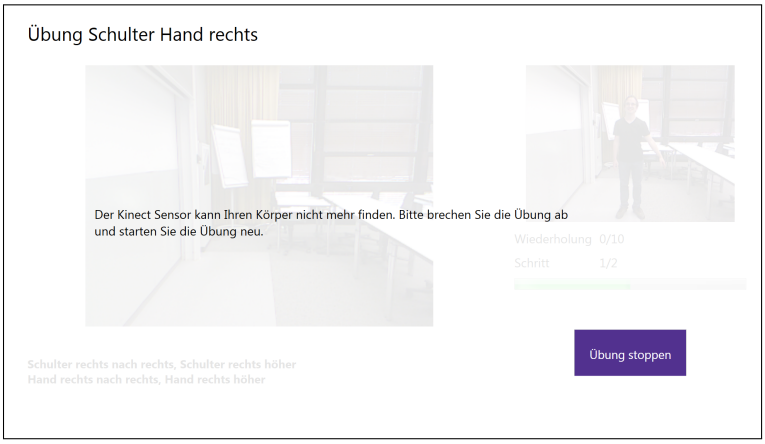


Abbildung 4.34: Screenshot: Unterbrechung der Übungsdurchführung (Person nicht mehr erkannt)

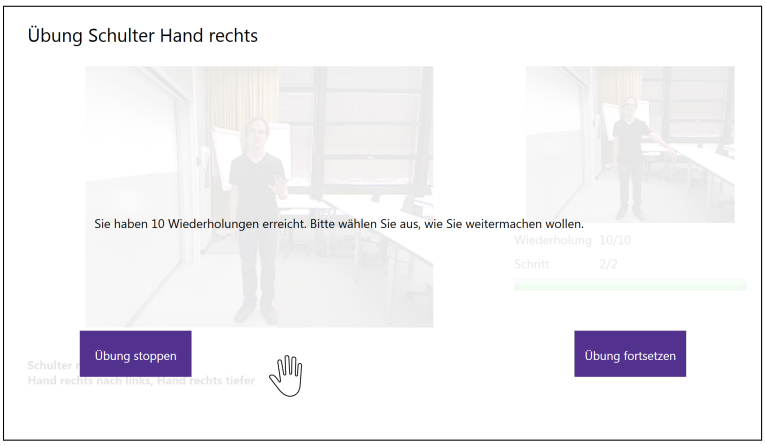


Abbildung 4.35: Screenshot: Unterbrechung der Übungsdurchführung (Anzahl Wiederholungen erreicht)

Die generierten Log-Dateien sind den jeweiligen Übung zugeordnet, um die Übersichtlichkeit der einzelnen Dateien zu erhalten. Eine Übersicht der vorhandenen Log-Dateien wird im selben Aufbau wie das Hauptmenü dargestellt (siehe Abbildung 4.36). Die Aus-

wertung von vergangenen Ausführungen kann so problemlos angesehen werden, siehe Abbildung 4.31, Button «Auswertungen ansehen».

Abbildung 4.36: Screenshot: Übersicht der vorhandenen Übungsauswertungen

Im Vergleich zum Prototyp wurde die Auswertung stark vereinfacht und beschränkt sich jetzt auf die Dauer einer Übungswiederholung (siehe Abbildung 4.37). Ziel ist nicht, den Benutzer unter Stress zu setzen. Er hat jedoch die Möglichkeit, seinen Fortschritt an Hand der durchschnittlichen Zeiten der einzelnen Durchführungen zu vergleichen.

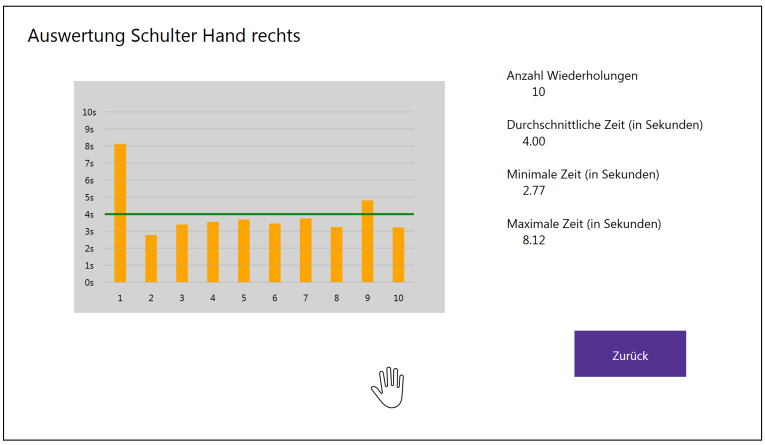


Abbildung 4.37: Screenshot: Auswertung einer Übungsdurchführung

Internationalisierung

Um die Applikation möglichst breit einsetzbar zu machen, sollten die Texte in Englisch und Deutsch erscheinen. Ebenso sollen die Text leicht anpassbar sein. Sind Texte hart im Code festgeschrieben, so ist es sehr mühsam, beispielsweise einen Tippfehler anzupassen. C# erlaubt es mit einer sogenannten «Ressourcen-Datei» zu arbeiten. Diese besteht aus einer Liste von Key-Value Paaren. Bezeichnungen für Buttons können ganz einfach in die

Liste eingetragen werden und später darauf zugreifen. Sämtliche Bezeichnungen können so zentral gespeichert werden, sodass redundante Bezeichnungen leicht gefunden werden.

Weiter besteht auch die Möglichkeit, einheitliche Bezeichnungen zu finden sowie eine mehrsprachige Applikation zu generieren. Hierfür wird die Ressourcen-Datei kopiert und auf die zusätzliche Sprache angepasst.

«KinectSensorChooserUI»

Bei weiteren Tests wurde erkannt, dass wenn keine Kinect angeschlossen ist, es unmöglich ist, die Applikation zu bedienen oder zu beenden. In einer vorhergehenden Entscheidung wurde das Schliessen mittels «Alt+F4» unterbunden. Dies, damit die Applikation sauber herunterfährt und die Kinect dabei ihre Sensoren ausschaltet.

Dadurch, dass die Applikation zwingend eine Kinect verlangt, muss im Fehlerfall eine Meldung ausgegeben werden. Ein Fehlerfall entsteht, wenn keine Kinect vorhanden oder diese fehlerhaft angeschlossen ist. Weiter soll die Ausführung der Applikation verhindert werden. Die Funktion, «KinectSensorChooserUI», wurde vom Beispiel des Kinect SDK übernommen und in die Applikation eingebaut.

Die Abbildung 4.38 zeigt die Fehlermeldung «Kinect wird benötigt», wenn keine Kinect angeschlossen ist.

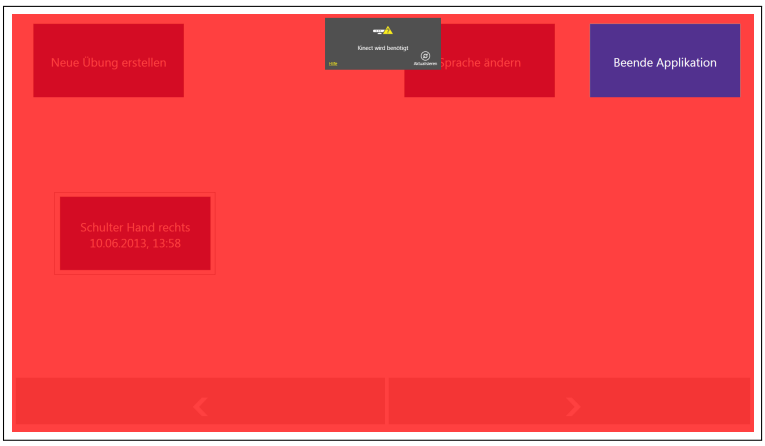


Abbildung 4.38: Screenshot: Fehlermeldung, dass keine Kinect angeschlossen ist

Fazit Implementation

Nach all diesen Anpassungen und Erweiterungen hat sich der Prototyp stark verändert und es entstand eine lauffähige Applikation. Die Softwarearchitektur ist im Kapitel Softwarearchitektur dokumentiert. Im Kapitel Schlussfolgerung und Ausblick wird auf die Umsetzung zurückgeblickt und ein Ausblick gemacht.

Kapitel 5

Softwarearchitektur

Dieses Kapitel umfasst die Abschnitte Anforderungsspezifikationen, Domainmodel, MVVM-Pattern und Namespaces. Unter anderem werden dabei die Aktoren und die Struktur der Applikation aufgezeigt.

5.1 Anforderungsspezifikationen

Bei der Erarbeitung der Anforderungsspezifikationen kamen die nachstehenden zwei Aktoren hervor:

- Patient oder Benutzer der Applikation
- Therapeut

Die zwei Aktoren unterscheiden sich insofern, dass sie die Applikation unterschiedlich anwenden. Diese Unterteilung ist in der Abbildung 5.1 ersichtlich.

5.1.1 Use Case Diagramm

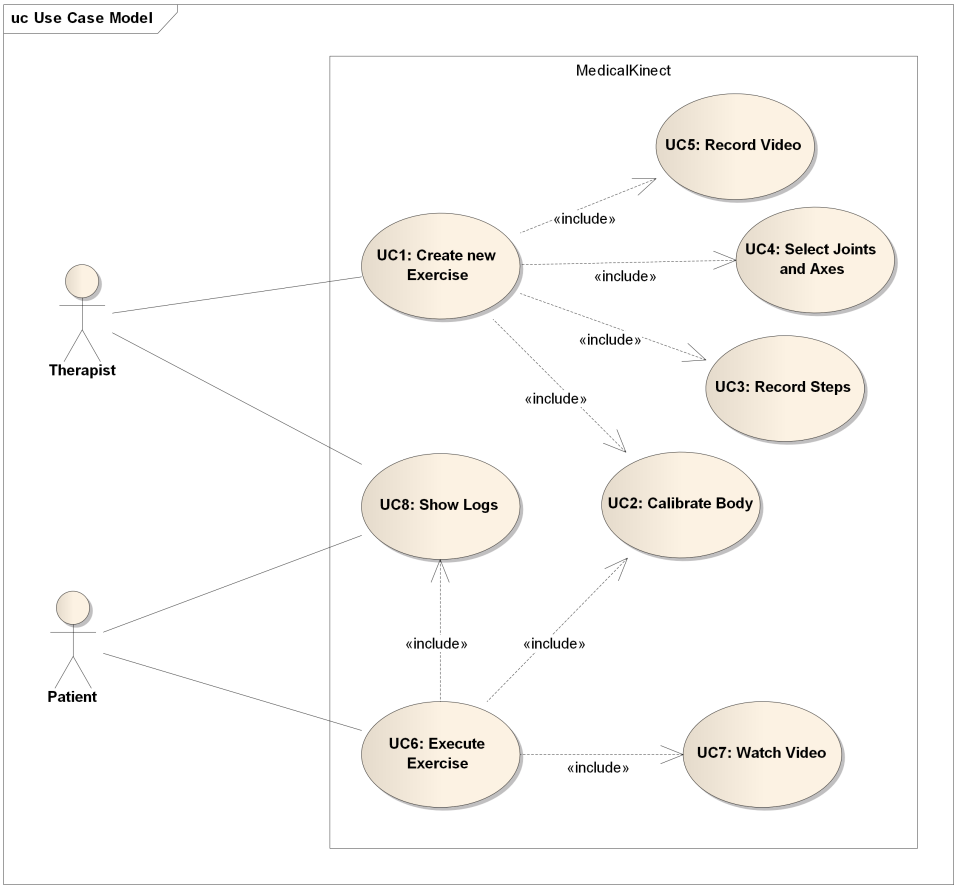


Abbildung 5.1: Use Case Diagramm

Die im Diagramm abgebildeten Use Cases werden wie folgt beschrieben:

- **UC1: Create new Exercise**
  Der Therapeut wählt im Hauptmenü «Create Exercise» an. Dadurch erstellt er eine neue Übung. Dazu gehören die Use Cases UC2, UC3, UC4 und UC5.
- **UC2: Calibrate Body**
  Der Therapeut steht vor der Kinect und hat sich entschieden eine neue Übung zu erstellen. Dafür muss nun zuerst sein Körper kalibriert und damit ausgemessen werden.
- **UC3: Record Steps**
  Nach der Kalibrierung des Körpers (siehe «UC2») muss der Therapeut die von ihm für die neue Übung vorgesehenen Schritte aufzeichnen.
- **UC4: Select Joints and Axes**
  Zum Schluss muss sich der Therapeut entscheiden, welche Gelenke und Bewegungsrichtungen für die Übung relevant sind und dadurch auch überprüft werden sollen.

- **UC5: Record Video**
  Damit der Benutzer die Übung besser verstehen kann, muss der Therapeut ein Video mit der Ausführung der Übung aufzeichnen.

- **UC6: Execute Exercise**
  Der Patient entscheidet sich eine Übung zu Therapiezwecken durchzuführen. Dafür wählt er im Hauptmenü die richtige Übung aus. Dazu gehören die Use Cases UC7, UC2 und UC8.

- **UC7: Watch Video**
  Damit der Benutzer die Therapie richtig ausführt, muss er zwingend zuerst das Video anschauen, welches der Therapeut aufgezeichnet hat.

- **UC8: Show Logs**
  Sowohl der Therapeut wie auch der Patient haben die Möglichkeit, die Auswertung einer vergangenen Übungsdurchführung anzuzeigen. Dem Patient wird sie direkt nach der Übungsdurchführung angezeigt.

5.2 Domainmodel

Durch die Verwendung des MVVM-Patterns (siehe Abschnitt MVVM-Pattern) und der generischen Übungen (siehe Abschnitt Generische Position der Gelenke im Kapitel Konzeptionelle Überlegungen) musste das Domainmodel angepasst werden. Das Modell konnte gegenüber dem Domainmodel des Prototyp stark vereinfacht werden. Es ist nun kompakter und übersichtlicher.

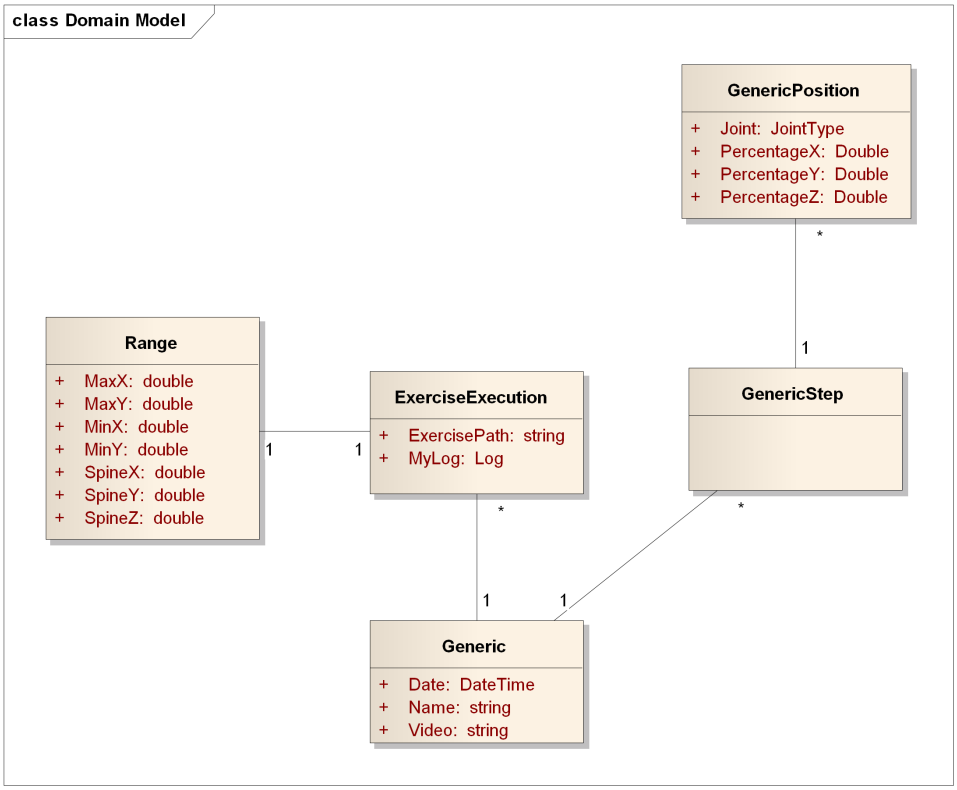


Abbildung 5.2: Domainmodel der entwickelten Applikation

In der Abbildung 5.2 ist die Klasse «ExerciseExecution» ersichtlich, welche eine Übungsdurchführung symbolisiert. Diese besteht aus einer generischen Übung («MyGeneric»), einem Log («MyLog») sowie dem Pfad («ExercisePath»), wo sich die Übung auf dem Computer befindet. Die generische Übung beinhaltet den Namen der Übung, das Datum, an welchem die Übung erstellt wurde, sowie der Name der Videodatei. Weiter beinhaltet die generische Übung eine Liste von «GenericStep's». Ein «GenericStep» bezeichnet dabei einen Teilschritt einer Übung. Dieser Teilschritt besteht aus einer Liste von «GenericPosition's». Informationen zu den Positionswerten eines Joints sowie der Name des Joints sind in diesem «GenericPosition»-Objekt gespeichert. Weiter ist für die Übungsdurchführung ein sogenannter «Range» notwendig. Diese Werte werden aufgrund der Kalibrierung des Benutzers gesetzt.

5.3 MVVM-Pattern

Die Software ist nach dem MVVM-Konzept aufgebaut. [24] MVVM steht für «Model View ViewModel». Dieses Konzept hat das Ziel, die grafische Oberfläche (das sogenannte User Interface) von der Businesslogik zu trennen und besteht aus drei Teilen:

- **Model**
  Bezeichnet die Klassen, welche die Businesslogik bilden. Die Inhalte können durch den Benutzer manipuliert werden.
  Beispiel: Adressverwaltung mit einer Klasse Person. Die Klasse Person beinhaltet typische Attribute (Daten) wie Name, Adresse und Geburtsdatum. Sie wird als Model angesehen.

- **View**
  Bezeichnet das User Interface. Die Eigenschaften der einzelnen GUI-Elemente binden sich mit denen des ViewModel.
  Beispiel: In der View sind einzelne Felder für die Anzeige der Personen-Attribute vorgesehen. Die Werte werden mit einem Property des ViewModel gebunden.

- **ViewModel**
  Das ViewModel ist die Schnittstelle zwischen der View und dem eigentlichen Model. Es beinhaltet ferner die UI-Logik.
  Beispiel: Eine Beispielview wird aufgerufen. Das ViewModel sorgt dafür, dass die konkreten Werte aus dem Model gelesen werden und in der View angezeigt werden. Erlaubt die View, Werte anzupassen und zu speichern, liest das ViewModel wiederum die Werte aus der View und speichert diese korrekt im Model. Eine anfällige Prüfung der Eingabewerte kann ebenfalls auf Ebene des ViewModels implementiert werden.

In C# besteht eine View aus einer «.xaml»-Datei sowie aus einer «.xaml.cs»-Datei. In der XAML-Datei wird dabei mit deklarativem XAML-Code das Aussehen des User Interfaces bestimmt. XAML ist die Kurzform für «Extensible Application Markup Language» und ist eine von Microsoft entwickelte Beschreibungssprache für grafische Oberflächen. [23] Die «.xaml.cs»-Datei beinhaltet den C#-Code, um beispielsweise das Verhalten nach einem Klick auf einen Button zu bestimmen, sofern das MVVM-Konzept nicht angewendet wird. Diese Code-Zeilen in der «.xaml.cs»-Datei werden als Code-Behind bezeichnet.

Wird das MVVM-Konzept umgesetzt, bestehen die Views aus dem deklarativen XAML Code (einer «.xaml»-Datei)) sowie wenigen Code-Zeilen im Code-Behind (in der «.xaml.cs»-Datei). Letzterer besitzt jedoch keine Logik. Im Code-Behind wird nur der DataContext gesetzt, welcher beim MVVM-Konzept ein ViewModel instanziiert.

Das ViewModel implementiert die Logik des User Interfaces und greift auf die Daten im Model zu. Die ViewModels erlauben jedoch auch, Unit-Tests durchzuführen. Wäre sämtliche Logik im Code-Behind, wären Unit-Tests nicht möglich, da die Logik nicht getrennt ist vom GUI.

5.4 Namespaces

Im Abschnitt 5.3 wurden die drei Teile des MVVM-Patterns erklärt. Diese Unterscheidung ist auch bei den Namespaces miteinbezogen worden. Zusätzlich wurden zwei weitere Namespaces verwendet. Die Abbildung 5.3 zeigt die Abhängigkeiten zwischen den einzelnen Namespaces.

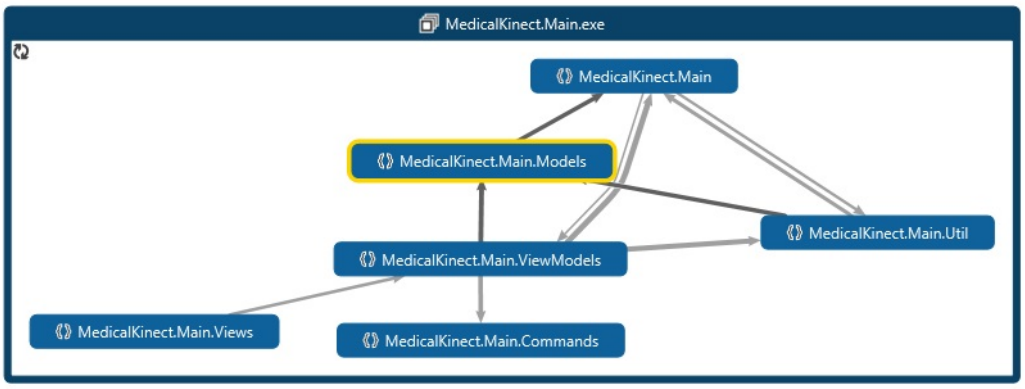


Abbildung 5.3: Dependency Graph (Abhängigkeiten zwischen den einzelnen Namespaces)

Die Namespaces können wie folgt beschrieben werden:

- **Models**
  Im Namespace «Models» befinden sich die Klassen mit der Businesslogik. Dies sind die Klassen, welche im Domainmodel enthalten und erklärt sind.
- **ViewModels**
  Die für die Kommunikation zwischen den Ansichten des User Interface und der Businesslogik verantwortlichen Klassen, den ViewModels, sind im gleichnamigen Namespace enthalten.
- **Views**
  Die XAML-Dateien, welche für die Ansicht des User Interface verantwortlich sind, werden im Namespace «Views» gespeichert.
- **Utils**
  Die für die Businesslogik benötigten Hilfsklassen werden als «Utils» bezeichnet.
- **Commands**
  Im Namespace «Commands» befindet sich die Klasse «RelayCommand». Diese implementiert das Interface «ICommand» und wird für die Buttons verwendet.

Kapitel 6

Usability-Tests

Während der ganzen Bachelorarbeitszeit wurden wiederholt Usability Tests durchgeführt. Sowohl aus eigenen Erkenntnissen wie auch aus solchen von Mitstudenten, konnte die Applikation ständig verbessert werden. Ausführliche Testprotokolle sind im Anhang Testprotokolle der Usability-Tests abgelegt.

6.1 Durchführung

Der Ablauf der Tests sollte wenn möglich alle Szenarien abdecken. Deshalb wurde von den Testpersonen verlangt, dass sie eine neue Übung erfassen und diese danach ausführen.

Abbildung 6.1: Testdurchführung

6.2 Allgemeine Erkenntnisse

Bereits zu Beginn war klar, dass die Kinect in der Genauigkeit eingeschränkt ist. Dies hat sich im Verlauf der Recherche bestätigt, wurde jedoch in Kauf genommen. Für allgemeine Übungen sollte die Genauigkeit ausreichen.

Generell hat sich dies auch bestätigt. Es ist jedoch zwingend notwendig, dass die Übung sehr genau aufgenommen wird. Sind beim Erstellen einer Übung nicht alle Gelenke sauber erfasst oder die Bewegungsunterschiede zu klein, so gleicht die Übungsdurchführung einem Glücksspiel.

Von einem Mehrspieler-Modus wurde in der Weiterentwicklung abgesehen, dies wurde auch von keiner Testpersonen vermisst. Ausserdem sollte die Applikation übersichtlich und nicht überladen wirken. Gestört wurden die Personen jedoch von Verzögerungen der Applikation, sodass im Verlauf der Tests ein leistungsfähiger Computer eingesetzt werden musste. Durch diese Erkenntnis erübrigte sich der Gedanke, die Anzeige bei der Übungsdurchführung nicht für jede Meldung anzupassen. Denn dies würde ebenfalls den Gedanken erwecken, dass der Computer langsam und überfordert ist.

Bei der Anzeige wurde weiter daran festgehalten, dass Farbbild und das Skelett anzuzeigen. Dies hat sich bei den Tests bewährt und das Skelett wurde nicht als störend empfunden.

6.3 Abschlusstest

Wie in der Aufgabenstellung beschrieben ist die Applikation unter anderem für Therapiepatienten geeignet. Glücklicherweise konnte dabei eine Testperson gefunden werden, die vor kurzem an der Schulter operiert wurde und aktuell in Therapie ist. Konkret führt die Person drei bis viermal pro Tag Übungen durch und geht einmal pro Woche zum Therapeuten. Ähnliche Übungen (beispielsweise mit einer Xbox) wurden noch nie gemacht. Das Testprotokoll ist im Anhang beigelegt.

Das Ziel dieses Tests bestand darin, dass die Testperson eine Übung erstellt und danach durchführt. Während dieser Zeit wurde beobachtet, wie die Testperson die Applikation bedient und wie gut verständlich diese ist.

Die Testperson hat eine konkrete Schultertherapieübung erfasst und diese durchgeführt. Zu Beginn war die Bedienung mit der Hand etwas ungewohnt, dies hat sich jedoch schnell geändert. Für das Erstellen einer neuen Übung brauchte die Person mehr Unterstützung, die verschiedenen Schritte der Applikation waren nicht sofort klar.

Da dies auch schon bei anderen Tests aufgefallen ist, wurde entschieden eine Anleitung zu erstellen (siehe Anhang Benutzerhandbuch). Vor der ersten Verwendung sollte diese gelesen werden. Ebenfalls wurden die bereits vorhandenen Beschreibungen in der Applikation überarbeitet. Es wurde darauf geachtet, dass diese so kurz und prägnant wie möglich sind. Lange Beschreibungen werden in den meisten Fälle überlesen.

Für die Testperson war es angenehm die Übungen durchzuführen. Direkte Hinweise haben die korrekte Durchführung unterstützt und erlauben eine bessere Kontrolle. Die Person glaubt weiter, dass diese Art der Therapie die Motivation steigern kann. Übungen können beliebig oft angesehen werden, damit die Sicherheit besteht, die Übung richtig durchzuführen.

Könnte die Person zusätzliche Funktionalität wünschen, so wäre dies eine sprachliche Ergänzung. So könnte beispielsweise die Anleitung zur Übung durch Sprache ergänzt werden. Ein Therapeut könnte die Übung dabei genau erklären. Entsprechende Hinweise könnten auch während der Durchführung abgegeben werden. Alternativ könnte währenddessen auch Musik abgespielt werden. Bedenken, von einer Kamera überwacht zu werden, hat die

Person nicht. Er kann sich jedoch vorstellen, dass andere Angst vor einer Aufnahme haben. Wäre die notwendige Hardware vorhanden, wäre eine Installation der Applikation kein Problem für ihn.

6.4 Fazit

Das Ergebnis der durchgeführten Tests ist positiv zu werten. Immer wieder kamen neue Erkenntnisse hervor, welche gut in die Arbeit eingebracht werden konnten. Ebenfalls hatten die Personen keinerlei bedenken, von einer Kamera überwacht zu werden, was sehr erfreulich ist.

Im Abschlusstest kam der Wunsch die Applikation sprachlich zu ergänzen. Ein Therapeut könnte zusätzlich zum Video die Übung erklären. Dies wurde jedoch vorher nicht bedacht, sodass dies im Rahmen dieser Bachelorarbeit nicht mit eingebaut werden konnte.

Kapitel 7

Schlussfolgerung und Ausblick

Im Verlauf der Bachelorarbeit konnte eine Applikation namens «Medical Kinect» entwickelt werden, welche die gewünschten Funktionen der Aufgabenstellung umsetzt. In dieser wurde eine Applikation gefordert, welche es mit Hilfe der Kinect erlaubt, Therapieübungen aufzunehmen sowie durchzuführen. Ebenfalls soll die Applikation einfach zu bedienen sein. Dies konnte mittels Usability Tests überprüft werden.

Konkret benötigt die Applikation einen Kinect-Sensor. Mit dem entsprechenden SDK kann auf die Daten der Kinect zugegriffen werden, sodass beispielsweise die Bedienung der Applikation mittels Hand möglich ist. Damit eine Therapieübung aufgezeichnet und durchgeführt werden kann, wird der sogenannte «SkeletonStream» der Kinect verwendet. Dieser liefert Informationen zu den erfassten Skeleton's und kann in der Applikation weiterverwendet werden. So können beispielsweise die Positionen der Gelenke herausgelesen werden, um Berechnungen mit den Werten zu tätigen.

Ebenfalls wurde der «ColorSteam» der Kinect verwendet. Dieser vereinfacht dem Benutzer das Verständnis für die gezeichneten Skelette. Zudem ist der «ColorStream» zwingend notwendig, um ein Video aufnehmen zu können.

Mit Hilfe der entwickelten Applikation können die aufgenommenen Übungen gut mit anderen Benutzer ausgetauscht werden. Alle notwendigen Informationen sind in zwei Dateien abgespeichert: einer XML-Datei, welche sämtliche Informationen zur Übung enthält, und einer AVI-Datei, welche das Video beinhaltet.

Rückblickend kann gesagt werden, dass die Aufgabenstellung einige Tücken mit sich brachte. Zum einen war dies die Aufgabe, eine Übung bei der Ausführung zu vergleichen. Dies war eine der grössten Herausforderungen der Arbeit. Zum anderen war es nicht ganz trivial eine intuitiv und einfach zu bedienende Applikation zu gestalten. Einige Schritte in der Übungserfassung sind nicht ganz einfach zu verstehen. Die Bedienung mit der Hand ist im ersten Moment auch eher gewöhnungsbedürftig. Dennoch konnte durch die erwähnten Usability Tests eine für die Benutzer angenehm bedienbare Applikation erarbeitet werden. Nach kurzer Zeit hatten die Testpersonen sich an die Bedienung gewöhnt.

Ausblick

Ob und inwiefern die Applikation in der Praxis eingesetzt wird, kann zum aktuellen Zeitpunkt nicht klar gesagt werden. Die Idee, welche in dieser Bachelorarbeit verfolgt wurde, hat bestimmt Zukunft und wird so oder in ähnlicher Form mit grosser Sicherheit demnächst zum Einsatz kommen. Microsoft verbessert den Kinect-Sensor und das SDK fortlaufend. Wann dieser Zeitpunkt sein wird, ist abhängig von den Therapeuten, den Patienten und den Versicherungsanstalten, welche für die Therapiekosten aufkommen oder eben auch nicht.

Generell kann die Arbeit der Therapeuten mit Hilfe dieser Anwendung erleichtert werden. Allgemeine Übungen müssen lediglich einmal aufgenommen werden und können an die Patienten verteilt werden. Patienten können diese Übungen bei sich zu Hause, in einer gewohnten Umgebung, durchführen und auch selber neue Übungen hinzufügen. In anderen Ländern wie beispielsweise den Vereinigten Staaten ist dies bereits zum jetzigen Zeitpunkt teilweise der Fall, dass Patienten die Therapie von zuhause aus machen.

Wie bei den Usability Tests festgestellt, können noch zusätzliche Funktionen eingebaut werden. So zum Beispiel eine sprachliche Ergänzung bei der Video Anleitung der Übung. Ebenfalls könnte bei der Übungsdurchführung ein Skelett angezeigt, welches die geforderten Positionen aufzeigt. Der Benutzer könnte sich an diesem orientieren.

Kapitel 8

Persönliche Berichte

Persönlicher Bericht Pascal Roman Artho

Erst gerade noch fand eine Vorbesprechung mit unserem Betreuer, Prof. Dr. Josef M. Joller, statt, um uns für das Thema zu bewerben. Jetzt, nach etwa 16 Wochen Arbeit, ist die Zeit schon vorbei und der Abgabetermin steht kurz bevor. Dabei blicke ich auf eine spannende und interessante Zeit zurück.

Bereits bei der Vorbesprechung war mir bewusst, dass Prof. Dr. Josef M. Joller zwar eine Idee besitzt, die Umsetzung für ihn jedoch noch völlig offen ist. Anhand von unseren Recherchen konnte die Arbeit ein wenig eingrenzt werden. Zusammen mit unserem Betreuer einigten wir uns, eine Applikation selbst zu entwickeln.

Als Grundlage hierfür diente uns das Software Developement Kit (kurz SDK) der Kinect. Microsoft stellt eine Vielzahl von Möglichkeiten in diesem SDK zu Verfügung und erlaubt es anhand von einfachen Beispielen, die Funktionalität der Kinect kennen zu lernen. Mir persönlich kam dies sehr zugute, da ich mich zuvor überwiegend mit Java befasst hatte und C# nur im Modul «Microsoft Technologien» kennen gelernt hatte.

Es zeigte sich schnell, dass die Kinect beliebig verwendet werden kann und nahezu alles irgendwie möglich ist. Um uns nicht in irgendeiner Art einzugrenzen, hat Prof. Dr. Josef M. Joller auf einen konkreten Industriepartner verzichtet. Er wollte und hatte uns sehr viel Spielraum gelassen, um selbst Ideen einzubringen.

Dieser erhaltene Spielraum war für uns Studenten etwas ungewohnt, sodass in den ersten Wochen nach der Recherche nicht ganz klar war, was genau angestrebt werden soll. Es handelte sich um ein Missverständnis zwischen Betreuer und Studenten. Als dieses beseitigt war und Prof. Dr. Josef M. Joller einen Prototyp gewünscht hat, war klar was wir zu tun hatten.

Trotz vielen Versuchen und dem erstellten Prototyp wurde die Applikation immer wieder umgestellt und angepasst. Immer wieder kamen neue Erkenntnisse hinzu, was geändert oder verbessert werden muss. Die Verbesserungen kamen

sowohl aus eigener Einsicht sowie von Rückmeldungen des Betreuers und Testpersonen. Diese Umstellungen sind jedoch aus meiner Sicht nicht negativ zu bewerten. Die Applikation konnte dadurch nur ausgereift werden.

Zusammen mit meiner Mitstudentin, Rebekka Zahler, konnten wir im Rahmen dieser Bachelorarbeit eine funktionsfähige Kinect-Anwendung erstellen. Da wir bereits die Studienarbeit zusammen geschrieben haben, kannten wir unsere Stärken und Schwächen. Die Arbeiten konnten so gut aufgeteilt und gemeinsame Entscheidungen getroffen werden. Die Arbeit mit ihr war sehr angenehm.

Fazit

Rückblickend bin ich mit der Themenwahl und der Betreuung durch Prof. Dr. Josef M. Joller sehr zufrieden. Ich durfte viel Neues kennenlernen und konnte einige Erkenntnisse sammeln, wie eine Kinect sinnvoll eingesetzt werden kann. Ebenfalls hat Prof. Dr. Josef M. Joller uns viel Freiraum gegeben, um selbst Ideen einzubringen, stand jedoch bei Fragen immer zu Verfügung und stellte die notwendige Infrastruktur bereit.

Weiter bin ich mir sicher, dass unsere Applikation eine gute Basis bildet und für den Einsatz im Gesundheitswesen gewappnet ist. Ob und inwiefern die Kinect mit dieser Applikation eingesetzt wird, kann ich nicht sagen. Ich bin mir sicher, dass der Grundgedanke dieser Bachelorarbeit sehr rasch vermehrt angestrebt wird. Insbesondere wenn man beachtet, dass die Kinect und die SDK ständig weiterentwickelt und verbessert werden.

Ich blicke auf eine sehr spannende und interessante Bachelorarbeit zurück und danke allen Beteiligten.

Pascal Roman Artho, Juni 2013

Persönlicher Bericht Rebekka Zahler

Bereits als uns Prof. Dr. Josef M. Joller von seiner Idee einer Medizinischen Anwendung mit der Kinect erzählt hat, war ich sofort begeistert von der Aufgabe. Nach einigen Besprechungen mit meinem Partner Pascal Roman Artho, wurden wir uns schnell einig, dass wir diese Arbeit in Angriff nehmen wollen.

Wir wussten bereits von Beginn an, dass wir sehr frei sein werden was die Ideen und Entwicklung der Arbeit anging. Die Studienarbeit die wir bereits zusammen geschrieben haben, hat viele Vorgaben beinhaltet, und wir konnten sehr wenig selber Entscheiden. Deshalb war dieses Projekt von Beginn an völlig anders. Wir konnten unsere Ideen selber entwickeln und ohne Rücksprache mit dem Dozent umsetzen.

Ich hatte allerdings meine Bedenken, da wir noch nicht genau wussten, was das Ziel der Arbeit ist. Dazu kam, dass unsere Kenntnisse in C#, noch nicht sehr ausgereift waren. Zusätzlich mussten wir zuerst herausfinden, welche Möglichkeiten die Kinect bietet. Es bestand die Gefahr, dass die Hardware für unsere Ideen nicht geeignet ist.

Für mich war die Bachelorarbeit eine Herausforderung. Da ich im bisherigen Studium gewohnt war, dass die Arbeiten und Projekte welche durchgeführt worden sind, immer sehr strickte Vorgaben beinhalteten, musste ich mich zuerst daran gewöhnen. Allerdings war es sehr angenehm zusammen mit Pascal Roman Artho Idee zu entwickeln und diese umzusetzen.

Die ungewohnte Situation, dass wir weitestgehend auf uns allein gestellt sind, hat uns zu Beginn noch Schwierigkeiten bereitet, allerdings konnte wir uns schnell an die neue Situation gewöhnen. Die Bedenken von meiner Seite verschwanden nach den ersten Wochen schnell. Wir haben uns sehr schnell in die neue Programmiersprache eingearbeitet und im Laufe des Projekts unsere Kenntnisse ständig erweitert und uns ein genaues Ziel der Arbeit setzen können.

Fazit

Rückblickend habe ich erkannt, dass ich sehr froh bin, eine Bachelorarbeit geschrieben zu haben, in welche ich frei von Vorgaben war. Wir waren selber verantwortlich, dass wir eine gute Arbeit mit guten Ideen entwickeln konnten. Dadurch habe ich nach der Studienarbeit eine komplett andere Erfahrung in Projekt- und Teamarbeit machen können.

Es war sehr spannend, nach der guten Zusammenarbeit in der Studienarbeit mit Pascal Roman Artho, auch die Bachelorarbeit mit ihm schreiben, denn die Zusammenarbeit mit ihm war stets sehr angenehm.

Für mich war die Bachelorarbeit ein voller Erfolg. Ich habe nicht nur meine Arbeit im Team verbessert, ich konnte mich auch fachlich weiterentwickeln.

Dazu kam die Erfahrungen, die ich sammeln konnte, in einer neue Situation zu sein, und die Ideen selber entwickeln zu können.

Rebekka Zahler, Juni 2013

Literaturverzeichnis

[1] Andreas B. Imhoff, Ralf D. Linke und René Baumgartner. *Checkliste Orthopädie - 2. Auflage*. Georg Thieme Verlag, Stuttgart, 2010.

[2] behindertenhilfe offenbach.de. Spielerischer Spracherwerb. http://behindertenhilfe-offenbach.de/index.php?id=178&tx_ttnews%5Btt_news%5D=268&cHash=9206f47659bbc949245473ac7677b888, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[3] chip.de. Wii und Co: Spiele helfen bei Schlaganfall-Therapie. http://www.chip.de/news/Wii-und-Co-Spiele-helfen-bei-Schlaganfall-Therapie_48591834.html, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[4] Microsoft Corporation. Human Interface Guidelines v1.7. http://download.microsoft.com/download/B/0/7/B070724E-52B4-4B1A-BD1B-05CC28D07899/Human_Interface_Guidelines_v1.7.0.pdf, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[5] Carl Franklin franklins.net. GesturePak for Kinect for Windows. http://www.franklins.net/gesturepak.aspx, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[6] Nick Garland. Kinect for XNA. http://kinectxna.blogspot.ch/, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[7] golem.de. Xbox-One-Sensor kommt 2014 auch für PC. http://www.golem.de/news/kinect-fuer-windows-xbox-one-sensor-kommt-2014-auch-fuer-pc-1305-99420.html, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[8] informationweek.com. Physical Therapy App Uses Microsoft Kinect. http://www.informationweek.com/healthcare/mobile-wireless/physical-therapy-app-uses-microsoft-kine/240008188, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[9] itechnews.net. Panasonic TH-47LF20, 47 Zoll Professional LCD Monitor. http://www.itechnews.net/wp-content/uploads/2010/06/Panasonic-TH-47LF20U-and-TH-42LF20U-Full-HD-LCD-Displays-for-Digital-Signage.jpg, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[10] ivoryegg.com. Fujitsu CELSIUS W380.
http://www.ivoryegg.com/imagesNew/210000/218895–4000606-8040.jpg, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[11] kinectotherapy.in. KinectoTherapy.
http://www.kinectotherapy.in/, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[12] lakesideautism.com. Technology - Lakeside Center for Autism.
http://www.lakesideautism.com/technology/, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[13] microsoft.com. DispatcherFrame.Continue-Eigenschaft.
http://msdn.microsoft.com/de-de/library/system.windows.threading.dispatcherframe.continue%28v=vs.110%29.aspx, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[14] microsoft.com. Face Tracking.
http://msdn.microsoft.com/en-us/library/jj130970.aspx, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[15] microsoft.com. Kinect for Windows Sensor Components and Specifications.
http://msdn.microsoft.com/en-us/library/jj131033.aspx, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[16] microsoft.com. MSDN - Microsoft Developer Network.
http://msdn.microsoft.com/, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[17] microsoft.com. Sensor Setup and Support.
http://www.microsoft.com/en-us/kinectforwindows/purchase/sensor_setup.aspx, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[18] Philippe Morier und Martin Weber. *3D-Gesture Recognition - Erkennen von 3D-Gesten mit Microsoft Kinect™*. Juni 2012.

[19] Philippe Morier und Martin Weber. *Gestik zur Computer-Interaktion - Erkennung von Gesten mit Hilfe von Computern*. Juni 2012.

[20] supportnet.de. Kinect for Xbox 360.
http://www.supportnet.de/articleimage/2391280/02-Windows-Version-der-Kinect-kommt-in-2012-Microsoft.png, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[21] The Hong Kong Polytechnic University. KineLabs: The New Era of Elderly Exercise and Stroke Rehabilitation.
http://myweb.polyu.edu.hk/k̃inelabs/, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[22] wikipedia.org. Euklidischer Abstand.
http://de.wikipedia.org/wiki/Euklidischer_Abstand, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[23] wikipedia.org. Extensible Application Markup Language.
http://de.wikipedia.org/wiki/Extensible_Application_Markup_Language, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[24] wikipedia.org. Model View ViewModel.
http://de.wikipedia.org/wiki/Model_View_ViewModel, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

[25] Carl Franklin youtube.com. GesturePak Gesture Recording/Recognition Toolkit for Kinect for Windows.
http://www.youtube.com/watch?v=TrGCAMmn1U0&hd=1, zuletzt besucht am 12. Juni 2013, 13:00 Uhr.

Abbildungsverzeichnis

Glossar

Animation-Units Vordefinierte und berechnete Werte beim FaceTracking. 23

Arnoud Icard Algorithmus zur Gestenerkennung. 13

Code-Behind Code-Zeilen in der .xaml.cs-Datei. 48

ColorFrame Entspricht einem Farbbild, welches von der Kinect aufgenommen wurde. 32

GenericPosition Abstrahiert eine Position eines gespeicherten Übungsschritt. 57

GenericStep Abstrahiert einen gespeicherten Übungsschritt. 57

GesturePak Bezeichnet das «Gesture Recording und Recognition Toolkit» von Carl Franklin. 16

grip Mitgeliefertes Feature der Kinect Interaction. Erlaubt es, mit der Hand zu scrollen. 19

GUI Graphical User Interface. (Grafische Benutzeroberfläche). 47

Johan Thelin Algorithmus zur Gestenerkennung, wird auch als «Recognizing Mouse Gestures» bezeichnet. 13

Joint Bezeichnet ein, durch die Kinect erkanntes, Gelenk. 20

Kinect Ein von Microsoft entwickelter Sensor. Er bietet die Möglichkeit, beispielsweise ein Spiel statt über ein Gamepad, mit der Gestik des ganzen Körpers zu steuern. 1

Kinect Interaction Feature-Set, welches mit der Kinect SDK mitgeliefert wird. 19

MVVM-Konzept Model-View-ViewModel Konzept. Trennt die grafische Benutzeroberfläche von der Businesslogik. 49

NonTracked Möglicher Status eines Skeleton's, sofern das Skeleton nicht gefunden wird. 22

ObservableList Wird für das Hauptmenü verwendet, um die Buttons dynamisch anzuzeigen. 49

One Dollar Algorithmus zur Gestenerkennung, wird auch als $1 und «Unistroke Recognizer» bezeichnet. 13

Per Ola Kristensson Algorithmus zur Gestenerkennung, wird auch als «Continous Gesture Recognition» bezeichnet. 13

PositionOnly Möglicher Status eines Skeleton's, sofern das Skeleton nicht vollumfänglich erkannt wurde, Position der Person aber erkannt wird. 22

push Mitgeliefertes Feature der Kinect Interaction. Erlaubt es mit der Hand einen Button anzuwählen. 19

Range Bezeichnet die Klasse, um Angaben des virtuellen Quader um die Person abzuspeichern. 67

SDK Software Developement Kit. 18

Seated Mode Kinect-Modus, bei welchem lediglich der Oberkörper betrachtet wird. 27

SensorColorFrameReady Event, welches aufgerufen wird, wenn die Kinect ein neues Bild aufgenommen hat. 32

SensorSkeletonFrameReady Event, welches aufgerufen wird, wenn die Kinect ein Skeleton erkennt hat. 32

Skeleton Bezeichnet die Menge aller Gelenke, welche die Kinect erfassen kann. 20

SkeletonFrame Buffer für Skeleton-Daten, welche vom Kinect-Sensor erkannt werden. 21

SkeletonPoint Beinhaltet die Positionsangaben in X-, Y- und Z-Achse. 20

SkeletonStream Muss aktiviert werden, damit die Kinect Personen erkennt. 21

SkeletonTrackingState Bezeichnet den Status des gefundenen Skeleton's. 22

Tracked Möglicher Status eines Skeleton's, sofern das Skeleton vollumfänglich erkannt wurde. 22

tracken Kinect erkennt die Bewegungen einer Person. 19

UI-Thread Bezeichnet den Thread, welcher die grafischen Benutzerobjekte wie beispielsweise Buttons erstellt. 46

User Interface Benutzeroberfläche. 29

XAML Extensible Application Markup Language. Wird für das grafische Design der Applikation verwendet. 68

XML Extensible Markup Language. 2

Anhang A

Verwendete Tools

A.1 Entwicklungsumgebung

Der C#-Code wurde im Visual Studio 2012 von Microsoft erstellt. Um die Kinect anzusteuern wurde das Kinect SDK v1.7 verwendet, welches einige Beispielanwendungen beinhaltet. Der Code wurde mit Hilfe des «JetBrains ReSharper 7.1.3» optimiert.

Für die Generierung des Videos wurde die «AviFile-Library» von «Code Project»[1] verwendet. Diese erlaubt aus Einzelbildern ein Video zu generieren.

A.2 Dokumentation

Für die Dokumentation der Bachelorarbeit wurde Latex benutzt. Um die Dateien zu bearbeiten, wurde MikTex 2.9 verwendet. Der Grund für die Wahl von Latex lag darin, dass sich diese Lösung gut für Gruppenarbeiten eignet. Weiter wurde Latex bereits in der Studienarbeit verwendet, sodass die Funktionsweise bekannt ist.

A.3 Grafik

Die Mockups wurden mit Balsamig Mockup in der Version 2.2.3 erstellt. Mit Enterprise Architekt 9 wurde das Domain Model erarbeitet und Microsoft Visio wurde für die Erzeugung von Grafiken verwendet.

Die Wahl der Tools fiel aufgrund der guten Erfahrungen, welche während dem Studium damit gemacht wurden.

A.4 Versionsverwaltung und Projektmanagement

Für die Arbeit wurde ein von der HSR zu Verfügung gestellte virtuelle Server benutzt. Darauf wurde SVN (Apache Subversion) für die Versionsverwaltung und Redmine für das Projektmanagement installiert.

[1] http://www.codeproject.com/Articles/7388/A-Simple-C-Wrapper-for-the-AviFile-Library

Anhang B

Installationsanleitung

B.1 Systemanforderungen

Damit der Kinect-Sensor funktioniert, wird ein Computer mit folgenden Hardwareanforderungen benötigt [17]:

- Windows 7, Windows 8, Windows Embedded Standard 7 oder Windows Embedded POSReady 7
- 32 bit (x86) oder 64 bit (x64) Prozessor
- Dual-core 2.66-GHz oder schnellerer Prozessor
- dedizierter USB 2.0 Anschluss
- 2 GB RAM

In den verschiedenen Tests hat sich gezeigt, dass diese Werte sicher nicht unterschritten werden sollten. Zusätzlich sind weitere Komponenten notwendig. Zum einen wird ein Kinect-Sensor benötigt («Kinect for Windows» oder «Kinect for Xbox»). Zum anderen ist ein grosser Bildschirm notwendig, welcher die Full-HD Auflösung unterstützt.

Auf dem Computer muss zusätzlich die «Kinect for Windows Runtime v1.7» beziehungsweise die «Kinect for Windows SDK v1.7» installiert sein, sodass der Treiber für die Kinect erfolgreich geladen werden kann. Wird die «Kinect for Xbox» verwendet, so muss zwingend die «Kinect for Windows SDK v1.7» installiert sein. Mit der «Kinect for Windows» reicht die «Kinect for Windows Runtime v1.7» aus.

In der Bachelorarbeit wurden für die Tests nachstehende Komponenten verwendet:

- Kinect-Sensor
  Kinect for Xbox
- Bildschirm
  Panasonic TH-47LF20, 47 Zoll Professional LCD Monitor, welcher via HDMI am Displayport des Computers angehängt ist.

- Computer
  Hardware: Fujitsu CELSIUS W380 Workstation mit Intel Xeon CPU X3450 (2.66 GHz) und 8 GB RAM
  Software: Windows 7 (64 bit, Service Pack 1) und installierter «Kinect for Windows SDK v1.7»

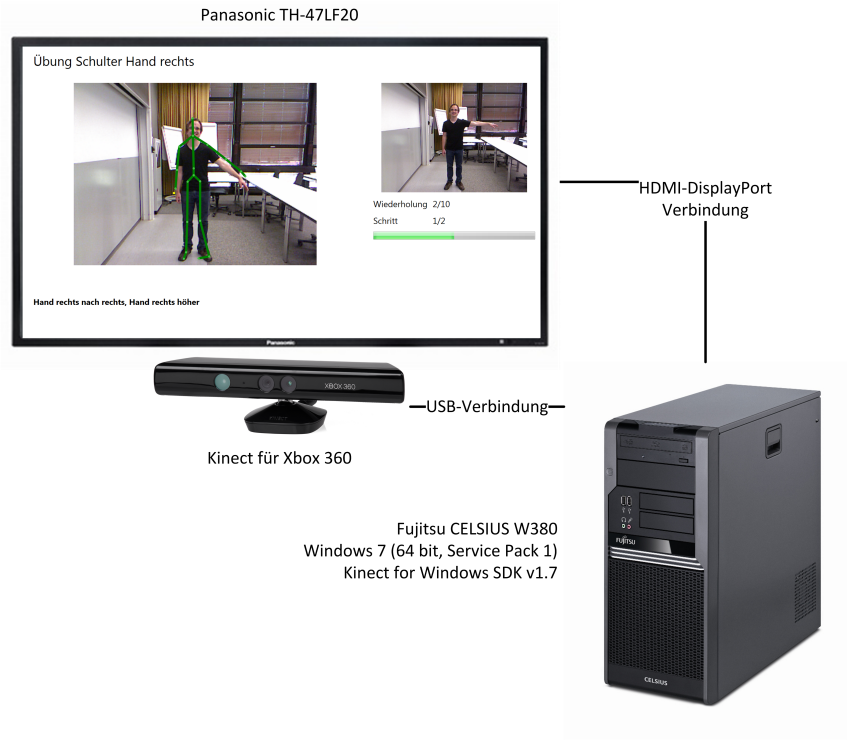


Abbildung B.1: Beispielaufbau «Medical Kinect» Applikation (Quelle der Bilder: Kinect [20], Bildschirm [9] und Computer [10])

B.2 Installation

Nach der erfolgreichen Installation der «Kinect for Windows Runtime v1.7»[1] beziehungsweise der «Kinect for Windows SDK v1.7»[2] kann die Kinect an den Computer angeschlossen werden. Nach dem alle Treiber erfolgreich installiert wurden, ist die Kinect einsatzbereit.

Die Anwendung dieser Bachelorarbeit besteht aus folgenden Dateien:

```
MedicalKinect/
├── de-DE/
│   ├── MedicalKinect.Main.resources.dll
│   └── Microsoft.Kinect.Toolkit.resources.dll
├── AviFile.dll
├── KinectInteraction170_32.dll
├── KinectInteraction170_64.dll
├── MedicalKinect.Main.exe
├── MedicalKinect.Main.exe.config
├── MedicalKinect.Main.pdb
├── MedicalKinect.Main.vshost.exe
├── MedicalKinect.Main.vshost.exe.config
├── MedicalKinect.Main.vshost.exe.manifest
├── Microsoft.Kinect.Toolkit.Controls.dll
├── Microsoft.Kinect.Toolkit.dll
├── Microsoft.Kinect.Toolkit.Interaction.dll
└── Microsoft.Kinect.Toolkit.pdb
```

Die Applikation kann mittels «MedicalKinect.Main.exe» gestartet werden. Beim Start wird, sofern notwendig, im Benutzerverzeichnis ein Ordner «.medicalKinect» erstellt (beispielsweise C:\Users\Peter Muster\.medicalKinect). In diesem werden sowohl Übungen inklusive deren Log-Dateien abgespeichert.

Achten Sie darauf, dass immer genügend Festplattenspeicher vorhanden ist! Für eine neue Übung werden bei der Erstellung bis zu 500 Megabyte benötigt.

[1] Download-Link: http://go.microsoft.com/fwlink/?LinkID=275590

[2] Download-Link: http://go.microsoft.com/fwlink/?LinkId=275588

B.3 Übungen austauschen und löschen

Wie bereits erwähnt werden sämtliche Übungen im Ordner «.medicalKinect» abgespeichert. Der Aufbau könnte wie folgt aussehen:

```
C:/Users/Peter Muster/.medicalKinect/
├── 20130610_120000/
│   ├── Logs/
│   ├── 20130610_120000.avi
│   └── 20130610_120000_Head_Exercise.xml
└── 20130610_121000/
    ├── Logs/
    ├── 20130610_121000.avi
    └── 20130610_121000_Shoulder_Right_Exercise.xml
```

In diesem Beispiel sind zwei Ordner vorhanden. Diese entsprechen je einer Übung. Soll eine Übung an jemand anderen weitergegeben werden, so kann der entsprechende Ordner kopiert werden. Die andere Person kann den Ordner entsprechend bei sich hineinkopieren. Das Löschen einer Übung funktioniert sinngemäss.
Mit dem nächsten Programmstart wird die Liste der verfügbaren Übungen angepasst. Sobald die Dateien nicht richtig gelesen werden können, steht die Übung in der Applikation nicht zu Verfügung.

Anhang C

Testprotokolle der Usability-Tests

Die nachstehenden Abbildungen zeigen die detaillierten Testprotokolle aus dem Kapitel Usability-Tests.

Test vom: 24.05.2013, 14:00

Testperson: Mirco Widmer

Neue Übung erstellen

Übung: rechtes Arm nach oben, linker nach unten

Axis: X, Y

Joints: Schulter, Ellenbogen

Sind die Beschreibungen gut verständlich?

Ja

Ist immer klar was zu tun ist?

Nein, bessere Beschreibungen

Wenn nein, wo nicht?

Schrittaufnahme

Ist der Aufbau der Applikation logisch?

Beim erstellen der Übung, Anzahl Schritte (Seite/Seite)

Ist die Bedienung der Applikation angenehm?

Button sind zu klein

Übung ausführen

Ist im jedem Schritt klar was zu tun ist?

Ja

Ist es angenehm die Übung auszuführen?

Ja

Weitere Bemerkungen:

Teilschrittaufnahme: sehen was aufgenommen wurde

Video generieren: Button nicht klar ersichtlich

Anzahl Durchführungen: 5/10 ← Gesamtzahl anzeigen

öfters Möglichkeit geben zurück ins Hauptmenü

Abbildung C.1: Test mit Mirco Widmer vom 24. Mai 2013

Test vom: 24.05.2013 14:25

Testperson: Cyrill Lam

Neue Übung erstellen

Übung: rechter Arm heben

Axis: Y

Joints: Schulter, Ellenbogen, Handgelenk (rechts)

Sind die Beschreibungen gut verständlich?

Nein, Teilschritt aufzeichnung nicht

Ist immer klar was zu tun ist?

Nein

Wenn nein, wo nicht?

Teilschritte ≠ Joints

Ist der Aufbau der Applikation logisch?

Ja

Ist die Bedienung der Applikation angenehm?

Buttons sind zu klein

Übung ausführen

Ist im jedem Schritt klar was zu tun ist?

Nein, siehe Bemerkungen

Ist es angenehm die Übung auszuführen?

Ja

Weitere Bemerkungen:

nicht klar, wann kalibrieren fertig ist und wann Übung beginnt

Abbildung C.2: Test mit Cyrill Lam vom 24. Mai 2013

Test vom: 5. Juni 2013, 17:45 Uhr

Testperson: Paul Fäh, Mitte März 2013 Schulter Operation

Neue Übung erstellen

Übung: Schulter bewegen

Axis: Z

Joints: Schulter rechts

Sind die Beschreibungen gut verständlich?

Ja

Ist immer klar was zu tun ist?

Mit Vorbehalt

Wenn nein, wo nicht?

Schritt-Aufnahme nicht sofort klar

Ist der Aufbau der Applikation logisch?

Ja

Ist die Bedienung der Applikation angenehm?

Zuerst etwas ungewohnt, aber man gewöhnt sich schnell daran

Übung ausführen

Ist im jedem Schritt klar was zu tun ist?

Anfangs nicht klar, was Log ist

Ist es angenehm die Übung auszuführen?

Ja, man bekommt sofort Feedback, was falsch ist

Weitere Bemerkungen:

statt Log, Übungsresultate

Abbildung C.3: Test mit Paul Fäh vom 5. Juni 2013

Allgemein

1. **Sind die Beschriftungen genug gross und leserlich?**

   Ja

2. **Bevorzugen Sie die Art dieser Übung (gegenüber der herkömmlichen Therapie)?**

   Es kann theoretisch zuhause angewendet werden, um spezielle Fälle abzudecken

3. **Wie oft führen Sie aktuell Therapieübungen durch?**

   3-4 / Tag
   einmal pro Woche Therapie bei Therapeut

4. **Haben Sie schon mal ähnliche Übungen gemacht (beispielsweise ein Xbox oder Nintendo Wii Spiel)? Falls ja, was ist an dieser Lösung besser / schlechter?**

   Nein, noch nie eine ähnliche Übung gemacht

5. **Würden Sie öfters Übungen durchführen, mit dieser Applikation?**

   Ja, diese Art der Therapie könnte mich motivieren, da direkte Rückmeldung

6. **Was finden Sie positiv an dieser Art der Therapie?**

   + Motivationssteigerung
   + Übung kann immer wieder angeschaut werden
   + bessere Kontrolle, da richtig / falsch angegeben wird

7. **Was finden Sie negativ an dieser Art der Therapie?**

   - besser die Anleitung zu Übung sprachlich Ergänzen
   - auch während der Übung sprachliche Hinweise
   - Musik während der Durchführung, wäre angenehm
   - evt. Vorbehalt durch "Aufnahme" der Kinectkamera

8. **Haben Sie zuhause den nötigen Platz um die Umgebung zu installieren? Würden Sie es installieren, falls Sie genug Platz hätten? (Falls nein, weshalb nicht?)**

   Keine Hardware vorhanden. Der Platz wäre ausreichend. Wenn Hardware da wäre, alles OK.

Abbildung C.4: Fragebogen zum Test mit Paul Fäh vom 5. Juni 2013

Anhang D

Projektmanagement

Meilensteine

Meilenstein 1: Recherche - 28. Februar 2013
In der zweiten Projektwoche wurden die gesammelten Ergebnisse der Recherche mit dem Betreuer besprochen. Aufgrund dieser Besprechung wurde der nächste Meilenstein festgelegt.

Meilenstein 2: Versuche mit der Kinect und Abgabe erster Dokumentationsentwurf - 27. März 2013
Im Rahmen des zweiten Meilenstein wurden Versuche mit der Kinect durchgeführt. Das Ziel bestand darin, die Kinect kennen zu lernen und die Möglichkeiten zu eruieren. Dabei wurde ein erster Entwurf der Dokumentation dem Betreuer zur Einsicht abgegeben. Der Entwurf bestand aus Recherchen, Grundlagen und ersten Erkenntnissen.

Meilenstein 3: Erster Prototyp - 11. April 2013
Ein erster Prototyp wurde dem Betreuer vorgestellt.

Meilenstein 4: erstes Release - 16. Mai 2013
Der Prototyp wurde verbessert und ein erster Release fertiggestellt.

Meilenstein 5: Abgabe - 14. Juni 2013
Der letzte Meilenstein ist die Abgabe der Bachelorarbeit.

Zeitplan

Für die Bachelorarbeit werden jedem Student zwölf ECTS angerechnet, dies bedeutet ein Aufwand von 360 Stunden pro Person. In der Abbildung D.1 und D.2 sind die geleisteten Stunden angegeben.

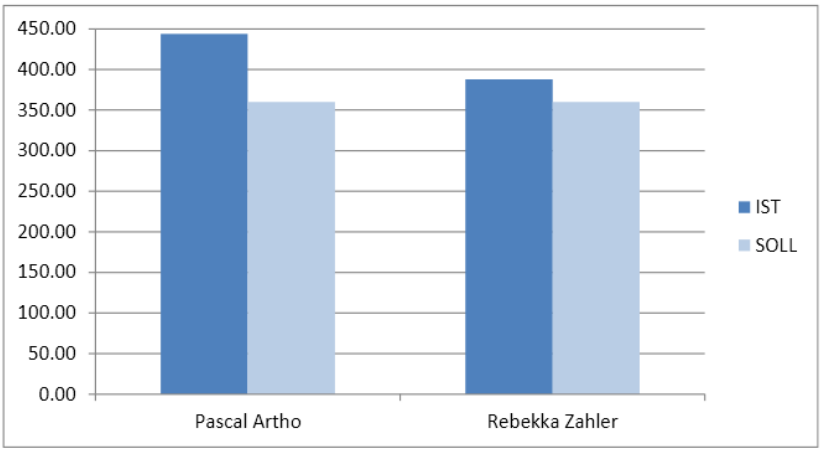


Abbildung D.1: Stundenverteilung Ist / Soll pro Person

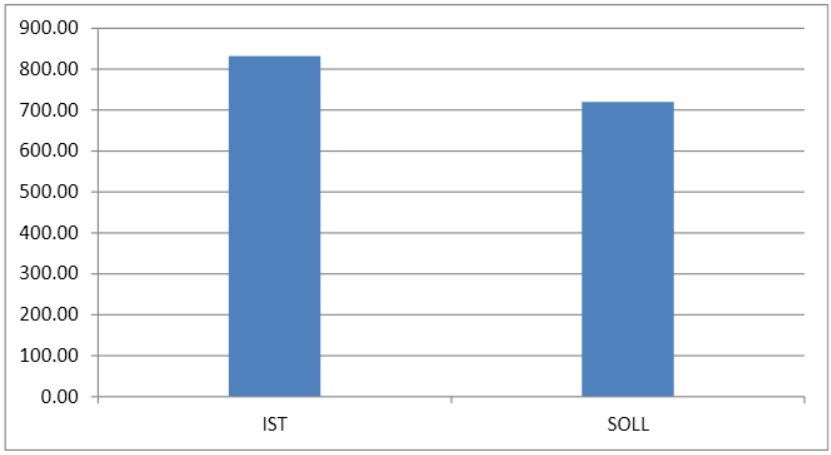


Abbildung D.2: Stundenverteilung Ist / Soll Gesamt

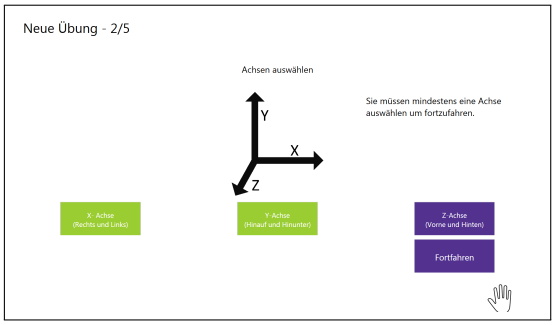
Anhang E

Benutzerhandbuch

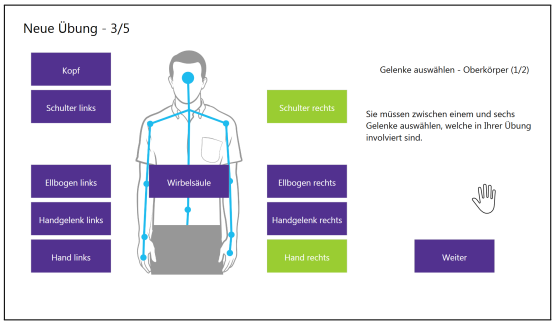
E.1 Übung erstellen

- Der Benutzer muss so vor dem Kinect-Sensor stehen, dass alle Gelenke wie im Bild angezeigt sichtbar sind. Falls dies nicht der Fall ist und der Körper nicht ganz im Bild ist, kann der Kippmotor der Kinect über die Buttons angesteuert werden. Dadurch verändert sich der Winkel der Kinect nach oben beziehungsweise unten.
- Erkennt der Kinect-Sensor mehrere Personen, so wird dieses gewählt, welches die kürzeste Distanz zum Kinect-Sensor aufweist.
- Bevor Teilschritte einer neuen Übung aufgenommen werden können, muss die Applikation den Benutzer kalibrieren und damit seinen Körper ausmessen. Dieser Prozess läuft nach einer Vorbereitungszeit von fünf Sekunden automatisiert ab.
- Nach dieser Kalibrierung und bei der Aufnahme der Schritte darf der Benutzer jeweils nur diejenigen Körperteile bewegen, welche relevant für die Übung sind. Die Position des Körpers darf sich nicht ändern, sofern dies nicht ein Teil der Übung ist.

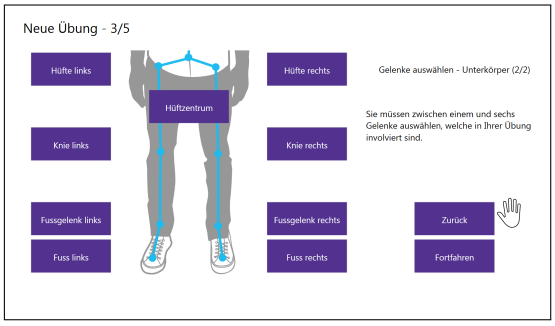
- Der Ablauf einer Übung besteht aus verschiedenen Schritten. Diese werden gespeichert und nachher bei der Ausführung verglichen. Sowohl die Ausgangsposition als auch die Endposition sollten gespeichert werden.
- Es ist wichtig, die Schritte so genau wie möglich aufzunehmen. Dazwischen sollte der Benutzer so wenig wie möglich bewegen, es sei denn, der aufzunehmende Schritt fordert es.

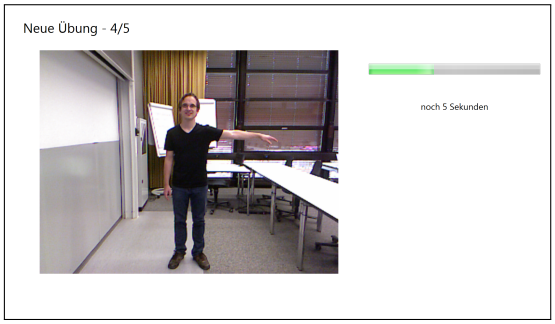


- Die Achsen entsprechen den Richtungen, welche überprüft werden sollen. Bewegt der Benutzer ein Gelenk bei einer neuen Übung nach oben und unten, entspricht dies der Y-Achse. Von rechts nach links betrifft es die X-Achse und von vorne nach hinten die Z-Achse.
- Der Benutzer muss mindestens eine Achse auswählen.



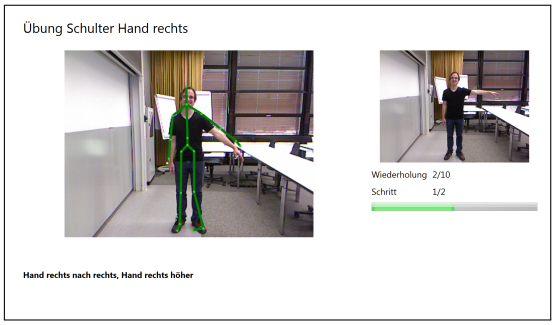
- Der Benutzer wählt aus total 19 Gelenken diese aus, welche mit dieser neuen Übung überprüft werden sollen.
- Maximal kann der Benutzer sechs Gelenke auswählen, welche in der Übung überprüft werden.



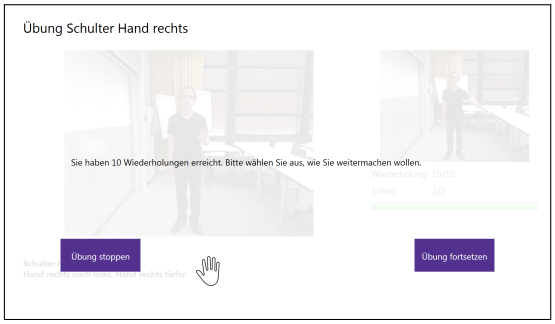


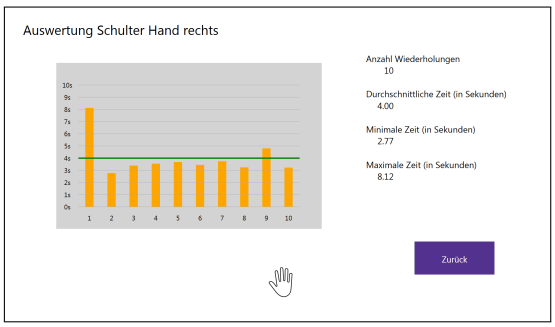
- Die Aufnahme und damit auch das Video dauert acht Sekunden. In dieser Zeit soll der Benutzer die Übung vorzeigen und vollständig durchführen.
- Das Video kann beliebig oft neu aufgenommen werden.

E.2 Übung durchführen



- Der Benutzer muss so vor dem Kinect-Sensor stehen, dass alle Gelenke wie im Bild angezeigt sichtbar sind. Falls dies nicht der Fall ist und der Körper nicht ganz im Bild ist, kann der Kippmotor der Kinect über die Buttons angesteuert werden. Dadurch verändert sich der Winkel der Kinect nach oben beziehungsweise unten.
- Erkennt der Kinect-Sensor mehrere Personen, so wird dieses gewählt, welches die kürzeste Distanz zum Kinect-Sensor aufweist.
- Bevor die Übung gestartet werden kann, muss die Applikation den Benutzer kalibrieren und damit seinen Körper ausmessen. Dieser Prozess läuft nach einer Vorbereitungszeit von fünf Sekunden automatisiert ab.
- Während der Übungsdurchführung sollte sich der Benutzer so wenig wie möglich bewegen, es sei denn, der aufzunehmende Schritt fordert es.





- Nach zehn Wiederholungen wird die Übung unterbrochen und der Benutzer kann auswählen, ob er mit der Übung weitermachen will. Will er diese Abbrechen, wird eine Auswertung angezeigt.
- Die Auswertung zeigt verschiedene Kennzahlen zur Durchführung, wie Anzahl Wiederholungen und die durchschnittliche Zeit einer Wiederholung.
- Vergangene Auswertungen können auch zu einem späteren Zeitpunkt betrachtet werden. Sie sind der Übung zugeordnet und so über das Hauptmenü erreichbar.